THE HOURGLASS EFFECT IN SOURCE-TARGET DEPENDENCY NETWORKS

A Dissertation
Presented to
The Academic Faculty

By

Kaeser M. Sabrin

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology

December 2018

THE HOURGLASS EFFECT IN SOURCE-TARGET DEPENDENCY NETWORKS

Approved by:

Dr. Constantine Dovrolis, Advisor
School of Computer Science
Georgia Institute of Technology

Dr. Bistra Dilkina
Department of Computer Science
University of Southern California

Dr. Ümit V. Çatalyürek
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Rahul C. Basole
School of Interactive Computing
Georgia Institute of Technology

Dr. Faryad Darabi Sahneh
Department of Mathematics
University of Arizona

Date Approved: October 31, 2018

To my family

ACKNOWLEDGEMENTS

I want to begin by thanking my advisor Professor Constantine Dovrolis. He has been a true mentor to me by providing constant guidance and support throughout this journey. The enthusiasm and unwavering passion he has for his research were motivational as well as educational for me, specially during the tough times in my Ph.D. pursuit. I am also grateful to him for continuously shaping my thought process to become more critical as well as effective, both necessary qualities to succeed in research.

I want to thank my committee members, Professor Bistra Dilkina, Professor Umit Catalyurek, Professor Rahul Basole, and Dr. Faryad Sahneh. I am grateful for the time they have spent in going through my work and for the valuable and interesting feedback they provided.

I spent a brief amount of time in the research labs of Professor Ling Liu and Professor Polo Chau. I want to thank them for giving me the opportunity to work and learn from them. I am grateful to Professor H. Venkateswaran for the guidance he has provided me throughout the doctoral pursuit.

A large part of my time at Georgia Tech was made enjoyable and fun by the many friends from Bangladesh Student community. I am grateful for their companionship that helped me to de-stress during the hardship of the Ph.D. life through many events, hangouts, and trips.

Family is the most important thing that kept me going through this pursuit. My parents are my constant source of love and support in everything I do. I am deeply thankful to my father-in-law, mother-in-law, and brothers-in-law for the continual encouragement and affection they gave me. The successful completion of my Ph.D. journey was only possible by the relentless motivation, support, and love given by my wife Lupu. This journey was equally hard for her as it was for me, and I am ever so grateful to her for the sacrifices she

endured and the patience she had shown despite that.

Lastly, my first child Sreyoshi Porsia Samin was born in the last year of my Ph.D. She is the greatest gift of my life and brightens each moment of my life with her thoughtful stares and laughs. Although she will not remember anything, she was a part of this journey and deserves a big thank from me for being so happily calm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Many hierarchically modular systems are structured in a way that resembles the shape of an hourglass: the system generates many outputs from many inputs through a relatively small number of intermediate modules that are critical for the operation of the entire system, referred to as the waist of the hourglass.

We first investigate the hourglass effect in hierarchical, but not necessarily layered, dependency networks. Our analysis focuses on the number of source-to-target dependency paths that traverse each vertex. We identify the core of a dependency network as the smallest set of vertices that collectively cover a given fraction of all dependency paths. We examine if a given network exhibits the hourglass property or not, comparing its core size with a "flat" (i.e., non-hierarchical) network that preserves the source dependencies of each target in the original network.

As a possible explanation for the hourglass effect, we propose the "Reuse Preference" (RP) model that captures the bias of new modules to reuse intermediate modules of similar complexity instead of connecting directly to sources or low-complexity modules.

We have applied this analysis in dependency networks that include technological, natural and information systems, showing that they exhibit the general hourglass property but to a varying degree and with different waist characteristics. We also compare the hourglass analysis framework with existing network "core finding" methods and compare path centrality with other vertex centrality metrics.

Finally, we extend our framework to networks that are not strictly hierarchical because they include feedback loops and lateral connections. In that context, we focus on the *C. elegans* brain network (connectome) and identify a core of ten neurons that almost all paths from sensory to motor neurons traverse. We explain the role of those neurons as

a dimensionality reduction mechanism, compressing the information provided by the 88 sensory neurons into a smaller set of intermediate-complexity functions that are re-used by the 119 motor neurons.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Complex systems in the natural, technological and information worlds are often hierarchically modular [78, 88, 94, 104]. A modular system consists of smaller sub-systems (modules) that, at least in the ideal case, can function independently of whether or how they are connected to other modules: each module receives inputs from the environment or from other modules to perform a certain function [9, 18, 121]. Modular systems are often also hierarchical, meaning that simpler modules are embedded in, or reused by, modules of higher complexity [93, 102, 105, 125]. In the technological world, modularity and hierarchy are often viewed as essential principles that provide benefits in terms of design effort (compared to "flat" or "monolithic" designs in which the entire system is a single module), development cost (design a module once, reuse it many times), and agility (upgrade, modify or replace modules without affecting the entire system) [48, 34, 81]. In the natural world, the benefits of modularity and hierarchy are often viewed in terms of evolvability (the ability to adapt and develop novel features can be accomplished with minor modifications in how existing modules are interconnected) [59, 60, 74] and robustness (the ability to maintain a certain function even when there are internal or external perturbations can be accomplished using available modules in different ways) [64, 65, 110]. In information sciences, hierarchical modularity can improve the stability, quality and speed of organizational search tasks (such as product or strategy development) [79, 116]. Additionally, it has been shown that both modularity and hierarchy can emerge naturally as long as there is an

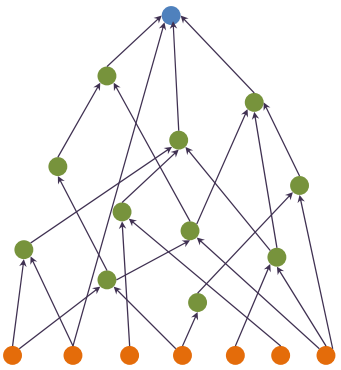
underlying cost for the connections between different system units [25, 77].

It has been observed across several disciplines that hierarchically modular systems are often structured in a way that resembles a bow-tie or hourglass (depending on whether that structure is viewed horizontally or vertically). Informally, this means that the system generates many outputs from many inputs through a relatively small number of intermediate modules, referred to as the "knot" of the bow-tie or the "waist" of the hourglass.[1] This "hourglass effect" has been observed in embryogenesis [20, 90], in metabolism [75, 113, 127], in immunology [12, 85], in signaling networks [111], in vision and cognition [91, 98], in deep neural networks [46], in computer networking [3], in manufacturing [112], as well as in the context of general core-periphery complex networks [28, 47].

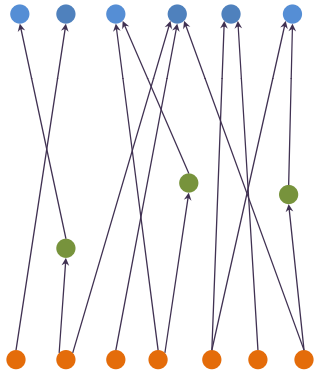
The few intermediate modules in the hourglass waist are critical for the operation of the entire system, and so they are also more conserved during the evolution of the system compared to modules that are closer to inputs or outputs [3, 29, 31]. These observations have emerged in a wide range of natural, technological and information disciplines, and so it is interesting to investigate whether the so-called *hourglass effect* has deeper and more general roots that are largely domain-independent.

In this dissertation, we present a quantitative framework for the investigation of the hourglass effect based on network analysis. First, the organization of a hierarchically modular system is transformed into a *dependency network*, i.e., a Directed Acyclic Graph (DAG) in which vertices represent either individual modules or Strongly Connected Components (SCCs) of interdependent modules. An edge from vertex u to vertex v in a dependency network means that module v depends, in a domain-specific manner, on module u. The input vertices of the dependency network are referred to as *Sources* and the outputs as *Targets*.

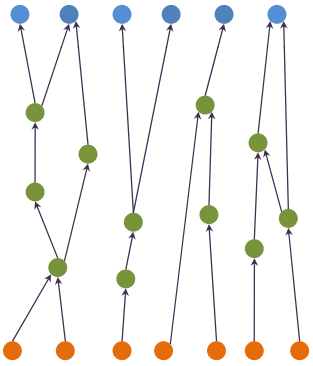
[1] The two terms, bow-tie and hourglass, have not been always viewed as synonymous in the network science literature. In particular, the term bow-tie has been applied even to networks for which the knot includes a large fraction of the network's vertices. We discuss the differences between the two terms in Section 1.2.



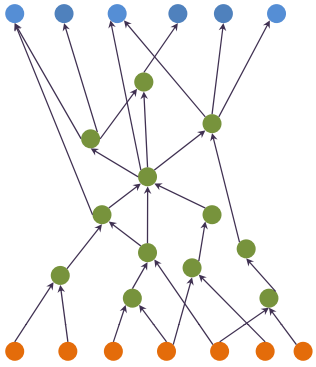
(a) Pyramid: few targets depend on many sources (or the opposite)



(b) Direct: targets often depend directly on sources, few intermediate vertices



(c) Decoupled: little reuse of common intermediate vertices across targets



(d) Hourglass: almost all source-target dependencies traverse a small number of intermediate vertices

Figure 1.1: Four toy examples of dependency networks with qualitatively different structure. The blue vertices are targets, the green are intermediates, and the orange are sources.

For example, four dependency networks with different qualitative structures are shown in Figure 1.1.

The importance of each vertex is quantified with a *path centrality* metric, defined as the number of source-to-target dependency paths that traverse that vertex. Based on that metric, we propose an algorithm to identify the *core* of the dependency network, i.e., the smallest set of vertices that collectively cover almost all (say 80-90%) of all source-to-target dependency paths. After computing the core, we can then evaluate if the given network exhibits the hourglass property or not by comparing its core size with a "flat" (i.e., non-hierarchical) network that preserves the source dependencies of each target. We also present a *Reuse Preference (RP)* model for the formation of a dependency network, capturing the bias of new modules to reuse intermediate modules of similar complexity instead of connecting directly to sources or low complexity modules.

We have applied this analysis framework in a diverse set of dependency networks from technological, natural and information systems: the call-graphs of two software systems, the metabolic networks of two species, and the citation networks of US Supreme Court cases for two legal matters (legality of abortion, and pension disputes). We show that these networks exhibit the hourglass property but to a varying degree. Further we quantify the location of the waist, relative to sources and targets, and the fraction of vertices in "tendril" paths that bypass the waist. The identified vertices at the waist of each network correspond to well-known important modules in the corresponding systems.

Ascribing relative importance to vertices and finding a set of key vertices in a network is a well-studied problem in the field of network analysis. We apply our hourglass framework as well as several other "core-finding" frameworks on real networks in a comparative study, which assesses the effectiveness of our framework in analyzing dependency networks. We also compare our path centrality metric with several more well known vertex

centrality metrics and show that it quantifies a different aspect of vertex importance than those metrics.

Next we extend our framework to analyze networks that are general digraphs containing many cycles. These networks cannot be transformed into DAGs without significantly distorting their topological structure. The cycles in these networks occur due to feedback loops and lateral connections and as such these networks are not strictly hierarchical. We employ domain-specific dependency path extraction strategies for these networks in order to apply our hourglass analysis framework on them. In this context, we focus on the *C. elegans* brain network (connectome) and identify a core of ten neurons that almost all paths from sensory (source) to motor (target) neurons traverse. We explain the role of those neurons as a dimensionality reduction mechanism, compressing the information provided by the 88 sensory neurons into a smaller set of intermediate-complexity functions that are re-used by the 119 motor neurons. Hierarchies in general digraphs (when present) are not easily identifiable as in directed acyclic graphs. We also address this problem by proposing a framework for inferring hierarchies in general digraphs utilizing dependency relationship information embedded in source-target paths.

Finally, we discuss the connections between the hourglass effect and related concepts such as the core-periphery structure of many complex networks, the presence of network bottlenecks, and the evolvability and robustness of systems that are hierarchically modular. Together with its theoretical significance, the hourglass effect may also have important practical value, especially in the design of technological systems that operate in uncertain or evolving environments.

1.2 Related work

The terms "hourglass" and "bow-tie" are often mentioned informally in the network science literature and in other disciplines – their precise meaning and whether the two terms are synonymous is not always clear however.

The term bow-tie, in particular, always refers to directed (but not necessarily acyclic) networks. It first appeared in the context of the WWW graph, after the 2000 study of Broder et al. [16]. The "knot" of that bow-tie was described as the largest Strongly Connected Component (SCC) in the graph, which included about 25% of the network's vertices. Similarly, the term bow-tie has been also used in the context of metabolic networks [75, 127]. Since then, several directed networks have been described as bow-ties, as long as there is a central SCC with incoming edges from a large input component and outgoing edges to a large output component [84, 19, 101, 120, 33]. In other words, the term "bow-tie network" refers mostly to a visual representation of directed networks based on the previous decomposition of vertices into four sets: an input component, a core (the largest SCC), an output component, and any other vertices that are not in the previous three components (referred to as "tendrils" and disconnected components). There is no requirement that the vertices in the knot of the bow-tie account for only a small fraction of the network size. There is also no requirement that the vertices in the knot are highly central, for any definition of centrality.

Hourglass networks, on the other hand, are typically directed and acyclic graphs, and the vertices at the hourglass waist need to be a small fraction of the total number of vertices. Further, the few vertices at the waist are present in almost all source-target paths in the network, and so they can be thought of as functionally very important for the underlying system [3, 29].

The three most relevant studies about the hourglass effect focused on the special case of layered and acyclic directed networks in which edges can only exist between successive layers [3, 4, 37]. In those studies, the hourglass effect is defined in terms of the number of vertices at each layer, and a network is referred to as an hourglass if the width of the intermediate layers is much smaller relative to the width of the input and output layers. The first study [3] proposed an evolutionary model (called EvoArch) for the emergence of the hourglass effect in computer networking protocol stacks; EvoArch captures the creation and competition between modules that perform similar functions and it may be also applicable in other layered technological systems. The second study [4] made the case that the topological structure of developmental regulatory networks (namely that the specificity of regulatory interactions increases during embryogenesis) is sufficient for the emergence of the hourglass effect in that context. The third study [37] showed that a layered and directed network can evolve to a bow-tie structure if the relation between inputs and outputs can be represented with a rank-deficient matrix, and if the mutations in the intensity (weights) of module interactions (network edges) can be modeled as products by a random number (rather than sums).

The previous models and analysis frameworks, however, are not applicable in more general dependency networks. Even if we artificially place vertices in layers based on topological sorting (i.e., sources are placed at the bottom layer, and each vertex is placed at the lowest possible layer so that all its incoming edges are from vertices of lower layers), edges can traverse more than one layer, and targets can appear at different layers. Additionally, a general dependency network may include cyclic dependencies and SCCs of interdependent modules. So, those studies do not define the hourglass property in general hierarchically modular systems and they do not how show how to identify their waist.

In the context of DAGs, a relevant prior study is [50]. That work had a different focus (not related to the hourglass effect or modeling hierarchical systems) but it considered the same

centrality metric (referred to as #P centrality) that we also use, and it analyzed the computational complexity of the problem of identifying the k vertices that have, collectively as a group, the largest #P centrality (referred to as the *C^3MC* problem in our work).

Another relevant study is the *BowTieBuilder* algorithm [111]. That work examined to what extent signal transduction pathways follow the bow-tie structure proposing a centrality metric ("bow-tie score") for each protein in the network, based on the number of sources and targets that are connected with paths traversing that protein. The knot of the bow-tie was defined as the set of proteins with maximal bow-tie score.

The "morphospace" of all possible hierarchical networks was investigated in [27]. The three dimensions of the considered morphospace in that study are "treeness", "feedforwardness" and "orderability". A large number of networks, mostly metabolic, neuronal and language, are shown to fall in the part of the morphospace that corresponds to hourglass or bow-tie networks.

1.3 Thesis outline

The structure of this dissertation is as follows:

- Chapter 2 has three key parts. Firstly, we present the general framework of analyzing the hourglass effect in a hierarchical, mostly acyclic dependency network. Secondly, we apply the hourglass framework to several real world dependency networks to examine the extent of their hourglass property. Finally, we present a comparative analysis using real networks between the hourglass framework and other classical network "core-finding" algorithms. We also make comparisons of the path centrality metric with other well known centrality metrics.

- In Chapter 3 we look into the question of "what causes the hourglass effect". We describe a model that demonstrates the emergence of the hourglass effect in networks that are synthetically generated but with realistic properties.

- We extend the hourglass analysis framework to handle networks that are highly cyclic in Chapter 4. We use the *C. elegans* brain network as an example of such a network and identify key neurons in that system. We provide two additional algorithms in this chapter. The first is for inferring hierarchy in general digraphs. The second quantifies the dimensionality reduction that is provided by an hourglass architecture.

- We summarize the dissertation contributions, provide a general discussion of this work, and propose possible extensions in Chapter 5.

CHAPTER 2

HOURGLASS EFFECT ANALYSIS FRAMEWORK

2.1 Dependency networks

Suppose that we are given a directed network $\mathbf{G_0}$ that represents a hierarchically modular system. Each vertex of $\mathbf{G_0}$ corresponds to a system module. An edge from vertex u to vertex v means that module v *depends on* module u. The precise meaning of this dependency relation is domain-specific. In a software system, for instance, modules may represent C functions and edges function calls (function v calls u). In a citation network, the modules may represent research papers or patents and edges some form of information transfer (v cites u). In a mechanical or chemical process, the modules may represent different devices or materials and the edges may represent that the construction (or composition) of a device (material) v requires u as input. Such hierarchical networks are ubiquitous across biology (e.g., food webs), technology (e.g., communication protocol stacks), organizations (e.g., reporting hierarchies), and information systems or social networks (e.g., meme propagation).

In general, the network $\mathbf{G_0}$ may include cyclic relations (referred to as "feedback loops", "recursive calls", etc, depending on the context) between two or more vertices. Each set of such interdependent modules can be identified as a Strongly Connected Component (an SCC is a set of vertices so that every vertex of that set can reach any other vertex of that set). In other words, the modules of an SCC do not have any hierarchical ordering between them; they are all mutually interdependent. To construct an acyclic hierarchical network, we first

compute all SCCs of $\mathbf{G_0}$; this can be done in linear time using Tarjan's algorithm [114]. Then, we replace every SCC of $\mathbf{G_0}$ with a single *super-vertex* that corresponds to the set of vertices in that SCC. Any incoming edge to a vertex of an SCC from a vertex outside that SCC becomes an incoming edge to the corresponding super-vertex; similarly, we construct the outgoing edges of each super-vertex from the outgoing edges of the corresponding SCC. The replacement of SCCs with super-vertices transforms the original network $\mathbf{G_0}$ into a Directed and Acyclic Graph $\mathbf{G}$. We refer to $\mathbf{G}$ as the *dependency network* that corresponds to the original network $\mathbf{G_0}$.

In the rest of the dissertation, the analysis will be focusing on dependency networks, and the notation will be as follows (Table A.1 in the Appendix lists all our notation). The *dependency network* $\mathbf{G}$ has a set $\mathbf{V}$ of vertices and a set $\mathbf{E}$ of directed edges. The number of vertices and edges is denoted by V and E, respectively.[1] The in-degree of v is denoted by $d_{in}(v)$ and the set of vertices that point to v is denoted by $I(v)$ (*inputs* of v). Similarly, the out-degree of v is denoted by $d_{out}(v)$ and the set of vertices that v points to is denoted by $O(v)$ (*outputs* of v). The *ancestors* of v is the set of vertices that can reach v, while the *descendants* of v is the set of vertices that v can reach.

The set $\mathbf{S}$ of vertices with zero in-degree are referred to as *Sources*, while the set $\mathbf{T}$ vertices with zero out-degree are referred to as *Targets*. The set $\mathbf{M}$ of remaining vertices represent *Intermediate* modules. We have that $\mathbf{V} = \mathbf{S} \cup \mathbf{T} \cup \mathbf{M}$. When plotting dependency networks, we follow the convention that sources appear at the bottom and targets at the top, and so edges have an upward direction.

A path $p(s, t)$ from a source s to a target t is referred to as a *source-target path*, or simply *ST-path*. Focusing on a target t, the set of all ST-paths that terminate at t represent the different "dependency chains" of sources and intermediates that are involved in the

[1] We denote the cardinality of a set $\mathbf{X}$ with X.

formation of t. We focus on all ST-paths that terminate at t instead of all source and intermediate vertices that t depends on. This distinction is important because a source or intermediate vertex v that participates in several ST-paths that terminate at t is more important for t than a vertex u that participates in fewer such ST-paths. For instance, in the context of a citation network the set of ST-paths that terminate at t represents all distinct ways in which the information contained in those source references has been transformed and propagated by intermediate references to finally produce t.

To quantify the topological importance of a vertex in a dependency network we rely on the following metric:

Definition 1 (Path Centrality)**.** *The path centrality $P(v)$ of a vertex v is the number of ST-paths that traverse v.*

This metric has been also referred to as the *stress* of a vertex [50]. Fig.2.1-a illustrates the path centrality of each vertex in a small dependency network.

$P(v)$ can be computed in $O(E)$ time, due to the acyclic nature of dependency networks. Suppose that $P_S(v)$ is the number of paths from any source to v, while $P_T(v)$ is the number of paths from v to any target. $P_S(v)$ can be computed in a bottom-up manner: $P_S(v) = 1$ for all sources and $P_S(v) = \sum_{u \in I(v)} P_S(u)$ for any v that is not a source. Similarly, $P_T(v)$ can be computed in a top-down manner: $P_T(v) = 1$ for all targets and $P_T(v) = \sum_{u \in O(v)} P_T(u)$ for any v that is not a target. It is easy to see that the path centrality of v is simply the product of $P_S(v)$ and $P_T(v)$,

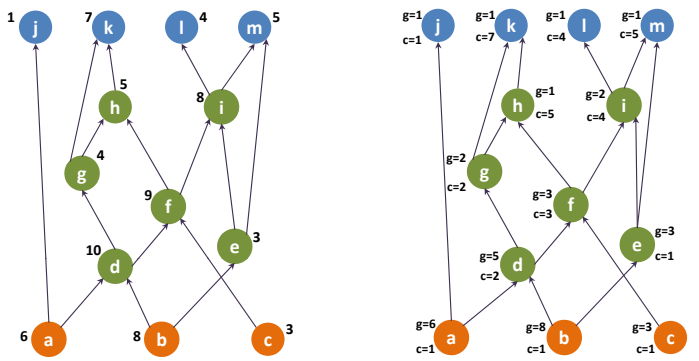
$$P(v) = P_S(v) \times P_T(v) \tag{2.1}$$

The path centrality metric can be also interpreted as follows. The number of paths $P_S(v)$ from sources to v can be thought of as a proxy for v's *complexity*: The more ST-paths

terminate at v, the more complex is the formation of v from all its ancestors. Sources have minimal complexity (set to one) because they do not depend on anything else. On the other hand, the number of paths $P_T(v)$ from v to targets can be thought of as a proxy for v's *generality*: The more ST-paths exist from v to the set of targets, the more general or common is the function provided by v in the formation of distinct targets. Targets have minimal generality (set to one) because they are not used to form any other module.

Based on these two definitions, *the path centrality of a vertex v is the product of v's complexity and generality.* This implies that path centrality is a metric that evaluates the topological importance of a vertex in both the upward and downward directions of a dependency network. If the complexity and generality of a vertex are both high, relative to other vertices, that vertex will also have high path centrality. Fig.2.1-b illustrates the complexity and generality of each vertex in a small dependency network.

The path centrality metric is more appropriate for identifying important vertices in a dependency network than other centrality metrics. The betweenness or closeness centrality metrics, for instance, only consider the shortest paths between two vertices, and so they would not assign high centrality to a vertex that participates in many (but relatively long) ST-paths. Also, the in-degree or out-degree of a vertex is a local metric and it does not capture the positioning of that vertex in the entire dependency network. The Katz centrality metric, on the other hand, does not distinguish between intermediate vertices and terminal (source or target) vertices, and it penalizes longer dependency paths. Some other centrality metrics, such as pagerank or eigenvector centrality [83], are not appropriate for DAGs.



(a) Path centrality (b) Complexity & generality

Figure 2.1: The path centrality of each vertex (shown at the left) and the generality (top number) and complexity (bottom number) of each vertex (shown at the right).

2.2 The core of a dependency network

Intuitively, the *core* of a dependency network can be defined as a subset of vertices that represent the most central modules in the underlying system. One approach would be to rank vertices in terms of path centrality. This approach does not consider however that two or more vertices may be traversed by almost the same set of ST-paths. So, even though they may both have high path centrality, including one of them in the core would be sufficient to "cover" those source-target dependencies.

Instead, we define the core of a dependency network based on the solution of an optimization problem: identify the smallest set of vertices that are traversed by almost all ST-paths – namely, a large fraction τ of all ST-paths. We approach this problem in two steps. First, we consider the problem of computing the most central set of k vertices, when k is given, which has already been studied by Ishakian et al. in [50]. Then, we use an algorithm for the previous problem to identify the minimum-size core for a given fraction τ of ST-paths.

Definition 2 (Coverage of ST-paths)**.** *Let* $\mathbf{P}$ *be the set of all ST-paths and* $\mathbf{R}$ *be a set of vertices.* $\mathbf{P_R}$ *is the subset of* $\mathbf{P}$ *that traverses at least one vertex in* $\mathbf{R}$*. The corresponding*

path coverage *of* $\mathbf{R}$ *is defined as:*

$$\delta_{\mathbf{R}} = \frac{P_R}{P} \tag{2.2}$$

Problem 1 (Cardinality-Constrained Core with Maximum Coverage – C^{3}MC)**.** *Given a cardinality* k*, identify a set* $\hat{\mathbf{R}}_k$ *of* k *vertices with maximum path coverage.*

$$\hat{\mathbf{R}}_k = \arg\max_{\mathbf{R}\subset\mathbf{V}:|\mathbf{R}|=k} \{\delta_{\mathbf{R}}\} \tag{2.3}$$

The set $\hat{\mathbf{R}}_k$ *may not be unique but* $\delta_{\mathbf{R}_k}$*, denoted as* $\hat{\delta}_k$ *in the following, is the same for all optimal solutions.*

The C^{3}MC problem is NP-Complete; a proof is given by Ishakian et al. [50] (they term it as the k-GCM(#P) problem). However, the objective function of the C^{3}MC problem is monotonically increasing (obvious) and submodular (see lemma A.2.1 in the appendix), and so the following greedy algorithm is guaranteed to produce an $(1 - \frac{1}{e})$-approximation of the optimal solution [82] – the same algorithm was also used in the work of Ishakian et al.

- Initially, the set $\hat{\mathbf{R}}_k$ is empty.
- In each iteration:
  1. Compute the path centrality of all vertices.
  2. Include the vertex with maximum path centrality in the set $\hat{\mathbf{R}}_k$, and remove it from the network (the case of ties is discussed in § 2.2.1).
- The algorithm terminates when the set $\hat{\mathbf{R}}_k$ includes k vertices.

The run-time complexity of the path centrality computation is $O(E)$ and, in the worst case, we need to recompute the path centrality of all vertices in every iteration of the algorithm. So, the run-time complexity of the previous greedy algorithm is $O(k\,E)$. In

Section 3.4, we show experimentally that the run-time of the core identification algorithm increases quadratically with the number of vertices N (if the average in-degree of non-source vertices remains constant).

2.2.1 Path centrality ties

We now describe how the previous greedy algorithm breaks ties among vertices that have the same maximum path centrality. Figure 2.2 illustrates that there are two different types of ties. First, it may happen that the tied vertices are traversed by exactly the same set of ST-paths. This will be the case, for instance, when those vertices are connected in a linear chain (vertices a, b and c in Figure 2.2). Whenever there is a maximum path centrality tie among a set of vertices that are traversed by the same set of ST-paths, we group these vertices as a single *Path-Equivalent Set* (PES). The elements of a PES are equivalent in the sense that they all capture the same set of ST-paths; it is sufficient to include any one of them in the core.

To identify a PES from a set of vertices that have equal path centrality, we pick a vertex u from that set and remove it from the network. Then, we recompute the path centrality of the remaining tied vertices and find those that now have zero path centrality. These vertices, together with u, form a PES. We repeat this process for any remaining tied vertices to identify additional PESs.

Second, there may be a maximum path centrality tie between two or more vertices that are traversed by different sets of ST-paths (for instance, vertices a and d in Figure 2.2). Ties of this type can be randomly broken, as long as it is sufficient to identify a single core instead of enumerating all possible cores. If it is necessary to identify all possible cores, we can consider separately every possible tie-breaker. This creates a tree of possible execution paths in which each leaf corresponds to a candidate core with k elements.

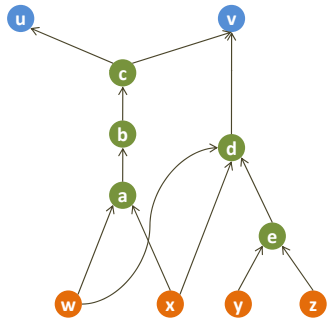


Figure 2.2: Vertices a, b, c have equal path centrality and they are traversed by the same set of ST-paths. Vertex d has equal path centrality but it is traversed by a different set of ST-paths. If there is a tie between the vertices a, b, c, d in an iteration of the core identification algorithm, either the first three vertices will be added in the core as a single Path-Equivalent Set (PES) representing the set $\{a, b, c\}$, or only vertex d will be added in the core.

2.2.2 The path coverage threshold τ

In practice, the cardinality of the core is not known a priori. Instead, we can set the cardinality of the core heuristically, as follows.

If it was required that the core is traversed by *all* ST-paths, the identification of the core would be equivalent to the well-known minimum-cut problem that can be solved efficiently with a max-flow algorithm [2]. However, requiring that the core covers every single ST-path is a very stringent condition; we have observed that in real dependency networks there are often some direct ST-paths that do not traverse any intermediate vertices or that do not share common intermediate vertices with most other ST-paths.

So, a more pragmatic definition is that the core of a dependency network should cover at least a fraction τ of all ST-paths, where τ is a given *path coverage threshold* that will typically be close to one. We define the problem of finding such a core as follows:

Problem 2 (τ-Core)**.** *Given a path coverage threshold τ, identify a set* $\mathbf{R}$ *of minimum*

cardinality so that $\delta_{\mathbf{R}} \geq \tau$.

$$\hat{\mathbf{R}} = \arg\min_{\mathbf{R} \subset \mathbf{V}: \delta_{\mathbf{R}} \geq \tau} \{|\mathbf{R}|\} \tag{2.4}$$

We prove that the τ-Core problem is NP-Complete (see Theorem A.3.1 in the appendix). To solve the τ-Core problem, we use the greedy algorithm for the C^{3}MC problem as follows: we iteratively run the algorithm for increasing cardinality (k) values starting with 1. For each value of k, we compute and store the solution set $\hat{R}_k$ and the corresponding maximum coverage $\delta_{\hat{R}_k}$ provided by the greedy algorithm. We stop after reaching a value of k for which all ST-paths are covered (i.e. maximum τ is reached). Then, for any given τ, we find the smallest k for which $\delta_{\hat{R}_k} \geq \tau$ and the corresponding $\hat{R}_k$ is our τ-Core.

We require a maximum of $O(E)$ operations for each vertex added to the solution to recompute the path centrality of the network and the number of vertices in the solution is dependent on the network. An upper-bound of the size of the solution set when the algorithm terminates is the minimum of the number of sources or targets, as that covers all possible ST-paths. In practice however, we expect the solution set's cardinality (and hence the resulting run-time complexity) to be much smaller than that.

We use the following notation to represent the core of a dependency network for a given τ: the set of vertices in the core is $\mathbf{C}(\tau)$, the size of the core is $C(\tau)$, and the path coverage of the core is $\delta_{\mathbf{C}(\tau)} \geq \tau$. Note that $\mathbf{C}(\tau)$ and $\delta_{\mathbf{C}(\tau)}$ may not be unique if there were ties during the computation of the core. The core size $C(\tau)$, however, is unique.

The increase of the path coverage of $\mathbf{C}$ when v is first included in that core is denoted by $\delta_{\mathbf{C}(v)}$. This metric also represents the *weight* of v in the core.

2.3 Hourglass dependency networks

2.3.1 Network flattening and H-score

Informally, the hourglass property of a dependency network can be defined as having a small core, even when the path coverage threshold τ is close to one. To make the previous definition more precise, we can compare the core size $C(\tau)$ of the given dependency network $\mathbf{G}$ with the core size of a derived dependency network that maintains the same set source-target dependencies of $\mathbf{G}$ but that is not an hourglass by construction.

To do so, we create a *flat dependency network* $\mathbf{G_f}$ from $\mathbf{G}$ as follows:

1. $\mathbf{G_f}$ has the same set of source and target vertices as $\mathbf{G}$ but it does not have any intermediate vertices.

2. For every ST-path from a source s to a target t in $\mathbf{G}$, we add a direct edge from s to t in $\mathbf{G_f}$. If there are w edges from s to t in $\mathbf{G_f}$, they can be replaced with a single edge of weight w.

Note that $\mathbf{G_f}$ preserves the source-target dependencies of $\mathbf{G}$: each target in $\mathbf{G_f}$ is constructed based on the same set of "source ingredients" as in $\mathbf{G}$. Additionally, the number of ST-paths in the original dependency network is equal to the number of paths in the weighted flat network (an edge of weight w counts as w paths). However, the dependency paths in $\mathbf{G_f}$ are direct, without forming any intermediate modules that could be reused across different targets. So, by construction, the flat network $\mathbf{G_f}$ cannot have the hourglass property.

Suppose that $C_f(\tau)$ represents the core size of the flat network $\mathbf{G_f}$. The core of $\mathbf{G_f}$ can include a combination of sources and targets, and it cannot be larger than either the set of sources or targets. Additionally, the core of the flat network is larger or equal than the core

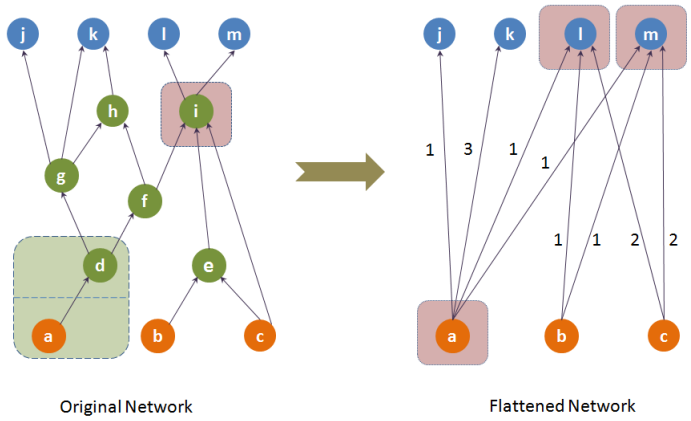


Figure 2.3: The weight of an edge in the flattened network represents the number of ST-paths between the corresponding source-target pair in the original dependency network. When the path coverage threshold is τ=90%, the core of the original network (left) is the set $\{\{a, d\}, i\}$ ($\{a, d\}$ form a *Path-Equivalent Set* and only one of them should be included in the core). The core of the flattened network (right) for the same τ is $\{a, l, m\}$. The H-score of the original network is $1 - \frac{2}{3} = 0.33$.

of the original network (because the core of the flat network also covers at least a fraction τ of the ST-paths of the original network – but the core of the original network may be smaller because it can also include intermediate vertices together with sources or targets). So,

$$C(\tau) \leq C_f(\tau) \leq \min\{S, T\} \tag{2.5}$$

To quantify the extent at which $\mathbf{G}$ exhibits the hourglass effect, we define the *Hourglass Score*, or *H-score*, as follows:

$$H(\tau) = 1 - \frac{C(\tau)}{C_f(\tau)} \tag{2.6}$$

Clearly, $0 \leq H(\tau) < 1$. The H-score of G is approximately one if the core size of the original network is negligible compared to the the core size of the corresponding flat network. Figure 2.3 illustrates the definition of this metric.

An ideal hourglass-like network would have a single intermediate vertex that is traversed

by every single ST-path (i.e., $C(1)$=1), and a large number of sources and targets none of which originates or terminates, respectively, a large fraction of ST-paths (i.e., a large value of $C_f(1)$). The H-score of this network would be approximately equal to one.

2.3.2 Coverage and location of a vertex

Another property of an ideal hourglass network is that all vertices that participate in ST-paths should be reachable from the waist, either in the upstream or in the downstream direction. To quantify this property, we define the *core vertex coverage* metric $U_{\mathbf{C}}$, where $\mathbf{C}$ is the core of the given dependency network:

$$U_{\mathbf{C}} = \frac{\sum_{v \in \mathbf{V_{ST}}} \phi_{\mathbf{C}}(v)}{V_{ST}} \tag{2.7}$$

where $\mathbf{V_{ST}}$ is the set of vertices that are present in one or more ST-paths, and $\phi_{\mathbf{C}}(v)$ is equal to one when v is a vertex that can reach, or that can be reached from, at least one vertex in the core $\mathbf{C}$; $\phi_{\mathbf{C}}(v)$ is zero otherwise. The metric $1 - U_{\mathbf{C}}$ can be thought of as the fraction of vertices in "tendril" paths that bypass the waist.

We can also associate a *location* with each vertex to capture its relative position in the dependency network between sources and targets. Computing the location of a vertex based on the *topological sorting* of the depending network would not be an appropriate approach in this context because that ordering is determined from the maximum distance of a vertex from the set of sources. Another way to place intermediate vertices between sources and targets is to consider the complexity $P_S(v)$ and generality $P_T(v)$ metrics that were defined in Section 2.1. Recall that sources have the lowest complexity value (equal to 1), while targets have the lowest generality value (equal to 1). The following equation defines a

location metric based on $P_S(v)$ and $P_T(v)$,

$$L(v) = \frac{P_S(v) - 1}{(P_S(v) - 1) + (P_T(v) - 1)} \tag{2.8}$$

$L(v)$ varies between 0 (for sources) and 1 (for targets). If there is a small number of paths from sources to a vertex v (low complexity) but a large number of paths from v to targets (high complexity), v's role in the network is more similar to sources than targets, and so its location should be closer to 0 than 1. The opposite is true for vertices that have high complexity but low generality – their location should be closer to 1 than 0.

We can also calculate an *average location for the entire core*. The weight of a core vertex v is proportional to the incremental increase $\delta_{\mathbf{C}(v)}$ of the path coverage of $\mathbf{C}$ when v was first included in that core. So, the average location of the core $\mathbf{C}$ can be defined as the following weighted average of the location of the core vertices,

$$L_{\mathbf{C}} = \frac{\sum_{v \in \mathbf{C}} [\delta_{\mathbf{C}(v)} \, L(v)]}{\sum_{v \in \mathbf{C}} \delta_{\mathbf{C}(v)}} \tag{2.9}$$

2.4 Case studies

In this section, we apply the previous analysis framework in six dependency networks from three different disciplines: two call-graphs (software engineering), two metabolic networks (biology, biochemistry) and two citation networks (information science). First, we present the corresponding datasets and the process to convert them into dependency networks. Table 2.1 shows the basic characteristics of the six dependency networks. Note that the networks vary considerably in terms of density, fraction of source or target vertices, and average ST-path length.

2.4.1 Datasets and dependency network construction

Call-graphs

Any non-trivial software system is written in a modular and hierarchical manner: "functions" (or "methods") are defined for distinct processing of tasks, and a function performs its task by calling other, simpler functions. The resulting hierarchy of function calls is referred to as the *call-graph* of that system. The sources of a software system are elementary functions that do not call any other function, functions provided by linked libraries, or functions that communicate directly with the primitives provided by the underlying hardware (e.g., device drivers) or the operating system. The targets are various applications or utilities that are called by external entities (the human user, other applications, libraries, systems, etc).

In the following, we analyze the call-graph of two complex and popular software systems: OpenSSH (version 5.2, written in C) and the Apache Math library (version 3.4, written in Java). The source code for OpenSSH was retrieved in a curated form from an earlier study [14] and the call-graph was constructed using CodeViz [40]. For the Apache Math library, we use the Java dependency graph extraction tool [41]. We follow the earlier convention that when a function v calls a function u, there is an edge from u to v.

In the case of OpenSSH, we first remove from the call-graph all functions that include the following keywords in their name: *main* (included in many C files for testing different parts of the system independently), *log* and *debug* (used during software development for debugging), *fail*, *fatal*, *error* (generic functions called in case of unexpected errors), and *exit* (program termination). The previous functions have high path centrality mostly because they are called by many other functions but they do not provide any information about the system architecture. Similarly, for the Apache Math library, we remove all *exception*

Table 2.1: Basic characteristics of analyzed dependency networks. All entries after the first row correspond to the Largest Weakly Connected Component (L-WCC).

	Networks					
	Software Call-graphs		Metabolic Networks		SCotUS Citation Networks	
Properties	*OpenSSH* *v-5.2*	*Apache Math* *v-3.4*	*Rat*	*Monkey*	*Abortion* *Cases*	*Pension* *Cases*
Vertices	1300	6685	843	845	1502	1290
Largest component (L-WCC)	99%	95%	64%	61%	100%	95%
Edges	4583	14823	612	588	3266	1555
Average degree	3.5	2.3	1.2	1.1	2.2	1.3
Targets	22%	35%	24%	25%	20%	24%
Intermediates	45%	32%	56%	55%	17%	11%
Sources	33%	33%	20%	20%	63%	65%
Average ST-path length	10.4	8.8	8.3	8.1	14.1	5.1
Number of super-vertices	3	24	10	9	0	0
Super-vertex size	2.5 ± 0.5	3.2 ± 4.1	9.4 ± 7.4	9.3 ± 7.2	-	-

handlers (methods associated with unexpected errors) and the methods of the *Object* class, which is the generic parent class for all Java programs.

The use of *recursive programming* (i.e., one or more functions forming a loop in the call-graph) creates cycles. As discussed in Section 2.1, each call-graph is transformed into a dependency network by first partitioning the call-graph in a set of SCCs, and then replacing each SCC with a single *super-vertex*. The number and size of the super-vertices in each call-graph are shown in Table 2.1.

Metabolic networks

Metabolic networks show how individual chemical reactions in the cell are combined to form the complex pathways associated with functions such as glycolysis or the biosynthesis of pyrimidine or purine [86]. There are large databases that provide reasonably accurate and complete metabolic networks for many species [56]. The KEGG database, in particular, has been curated for more than a decade to include all known metabolic reactions that conform with the available sequenced genome information [57].

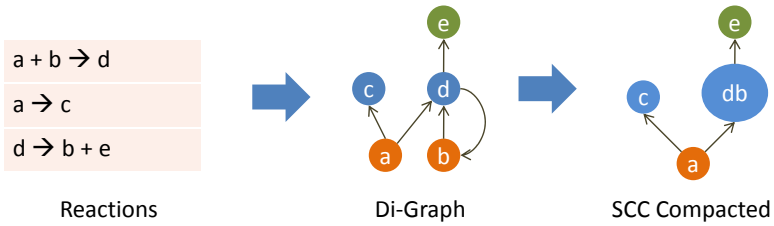


Figure 2.4: Construction of a dependency network from a given set of metabolic reactions.

In a metabolic network, the products of one chemical reaction can be used as substrates for another chemical reaction. This flow of matter and energy can be represented as a directed network where vertices correspond to metabolites, and an edge from u to v means that there is at least one reaction in which u is a substrate (input) and v is a product (output). Although most chemical reactions are reversible, most metabolic pathways are typically considered to flow in one direction. In the KEGG database, each reaction is associated with the most common direction in a given pathway.

A metabolic network often includes cycles. If two or more metabolites are present in the same cycle, it means that there is no hierarchical ordering between them – they are mutually interdependent. So, as in the case of call-graphs, after constructing the initial metabolic network we replace each SCC with a single *super-vertex* that represents the corresponding set of metabolites in that SCC. Figure 2.4 shows a small example of how a given set of chemical reactions can be first transformed to a directed network, and then to a dependency network.

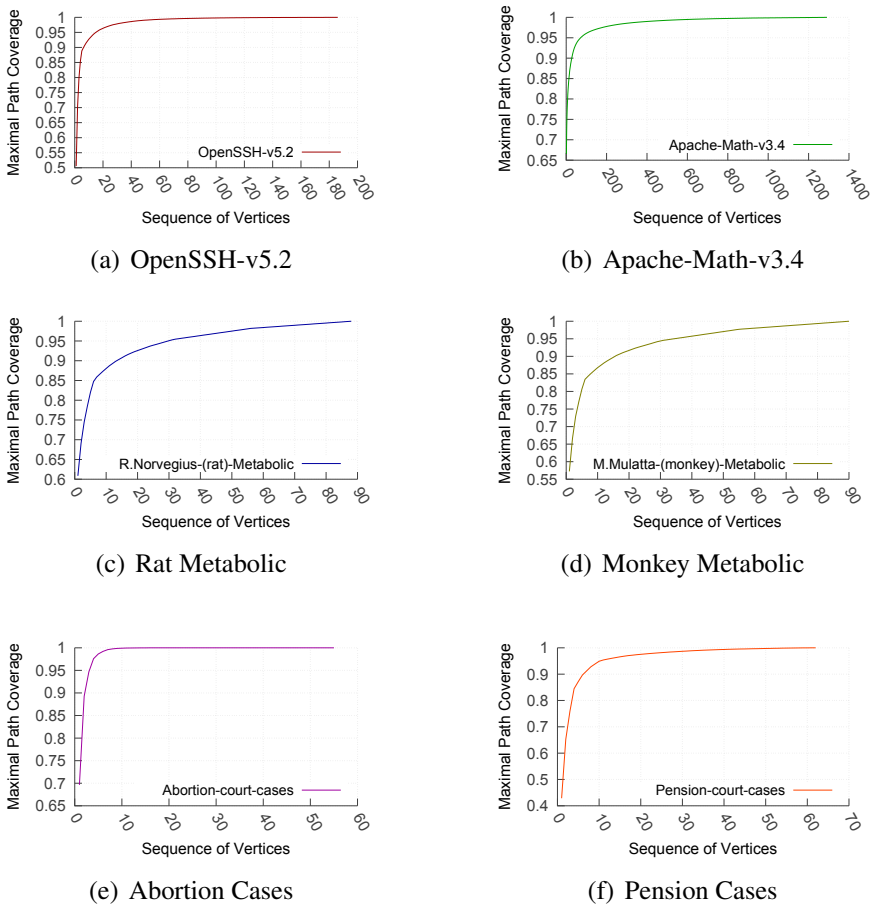
In the following, we present results for the metabolic networks of two organisms: *Rattus norvegius* (rat) and *Macaca mulatta* (monkey). Both datsets were retrieved from the 2014 KEGG [57] database. For each metabolic network we only analyze the Largest Weakly Connected Component (L-WCC). The smaller connected components correspond to distinct pathways that do not have any common metabolites with the L-WCC.

SCotUS citation network

Dependency networks can also capture the flow of information, knowledge or legal precedent in research publications, patents, court cases, and so on. Here, we focus on the citation network of court judgments made by the Supreme Court of the United States (SCotUS). We rely on a dataset collected by Fowler [36, 35] that includes all SCotUS cases between 1754 and 2002. Judicial decisions often leverage the precedent of earlier judgments to support their arguments, forming a directed citation network. Following our earlier convention, if a court case v refers to a previously settled case u, there is an edge from u to v. In the case of citation networks, the hierarchy of the dependency network implies a temporal ordering between connected vertices: if there is a path from u to v, u appeared before v.

In this study, we focus on two legal matters that have been the subject of many SCotUS cases: the *legality of abortion* and various *pension (or benefits) disputes*. First, we use the Legal Information Institute [71] of Cornell University's online legal library to find the set of SCotUS cases that focus on each of these two matters. Suppose that $\mathbf{X}$ is the set of SCotUS cases that are related to one of these two matters. We construct the corresponding citation network by including all cases in $\mathbf{X}$ as well as any other SCotUS case that directly cites, or is directly cited by, a case in $\mathbf{X}$. This expansion of the citation network with cases that do not belong in $\mathbf{X}$ is important because the SCotUS decisions about a certain matter may depend on, or they may have influenced, decisions regarding other legal matters.

The selection of sources and targets in a citation network may appear as somewhat arbitrary. This is an important issue that deserves further discussion. The sources and targets of a dependency network should be selected based on the scope, or boundaries, of the underlying system we aim to understand. Considering only parts of that system, or merging it with other systems, can mislead the analysis. For instance, if we want to identify the most significant publications associated with a specific problem in network science, say



(a) OpenSSH-v5.2 (b) Apache-Math-v3.4

(c) Rat Metabolic (d) Monkey Metabolic

(e) Abortion Cases (f) Pension Cases

Figure 2.5: The maximum path coverage $\hat{\delta}_k$ as a function of k for the six dependency networks.

community detection, it would be incorrect to only consider the citation network of publications that focus on spectral graph partitioning, and it would also be incorrect to consider every publication that relates broadly to graphs or networks. We admit, however, that in some cases it may be challenging to uniquely identify the scope, or boundaries, of a given dependency network; this is a problem that deserves further study.

The two citation networks are acyclic, and so we do not create any super-vertices.

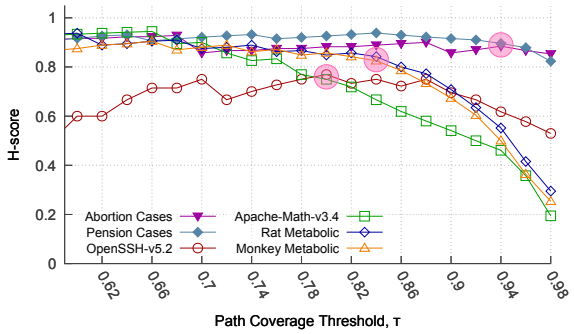


Figure 2.6: Effect of τ on H-score. The value of τ that we use in the rest of the analysis for each network is shown with a magnified symbol.

2.4.2 Analysis of dependency networks

Figure 2.5 shows the maximum path coverage $\hat{\delta}_k$ that results from solving the $\mathbf{C^3MC}$ problem iteratively, for increasing values of k, until $\hat{\delta}_k$ approaches 100%. Note that all six curves are strongly concave and that almost all ST-paths are covered with a very small number of vertices relative to the size of each network.

Figure 2.6 examines the effect of the path coverage threshold τ on the resulting H-score of each network. As expected, if we require that the core covers a higher fraction of ST-paths the core will need to be larger. The two citation networks strongly exhibit the hourglass effect, as their H-score remains close to 0.9 even when the core covers 90-95% of all ST-paths. The two metabolic networks can be also viewed as hourglass networks, with an H-score of about 0.85, but only as long as the core covers less than 80-85% of all ST-paths. Their core would need to be significantly larger to cover the remaining paths. The two call-graphs are structured differently and they exhibit a weaker hourglass effect: OpenSSH's H-score varies erratically between 0.6 to 0.8 depending on τ, while the Apache Math library's H-score quickly drops below 0.8 when the core needs to cover more than 80% of all ST-paths.

Based on Figure 2.6, in the rest of the analysis we set τ at the largest value before the

Table 2.2: Properties of the identified core for each dependency network.

Core Properties	Networks					
	Software Call-graphs		Metabolic Nets		SCotUS Citation Nets	
	OpenSSH v-5.2	*Apache Math v-3.4*	*Rat*	*Monkey*	*Abortion Cases*	*Pension Cases*
Path coverage threshold τ	0.8	0.8	0.85	0.85	0.95	0.95
Core size C	3	9	7	8	4	11
C/V	0.002	0.001	0.01	0.02	0.002	0.008
H-score	0.77	0.75	0.82	0.81	0.86	0.89
Number of distinct cores	1	1	1	1	1	1
SCCs in core	0	1	3	3	0	0
Number of PES in core	0	2	1	2	0	0
Core vertex coverage	0.35	0.21	0.53	0.57	0.82	0.48
Average core location	0.50	0.12	0.45	0.44	0.74	0.24

H-score shows a significant drop. After selecting the same value for each network type, we set τ as follows: call-graphs τ=80%, metabolic networks τ=85%, and citation networks τ=95%.

Table 2.2 summarizes the key properties of the core of each dependency network. The size of the core $\mathbf{C}$ varies from 0.1% to 1% of the network size V. In all six networks we identified only one core (no ties); some vertices in the core of the metabolic networks and of the Apache Math network are super-vertices. For the selected values of τ, the H-score is higher than 0.75 in all networks.

Even though the core of each network is quite small, relative to the total number of vertices, none of these networks can be described as an "ideal hourglass". This is shown both in terms of the H-score in Figure 2.6 and by the core vertex coverage: there is a significant fraction of vertices (about 20-80%, depending on the network) in ST-paths that bypass the core ("tendril paths"). The fraction $1 - \tau$ of ST-paths that bypass the core traverse at least two vertices each (a source and a target). When these tendrils traverse several intermediate vertices however, the core vertex coverage can be significantly lower than $2 \times (1 - \tau)$. As shown in the modeling results of the next section (see Figure 3.3-a), such low values of the core vertex coverage can be expected when each vertex has a bias to depend on vertices of similar complexity with itself (rather than to depend directly on

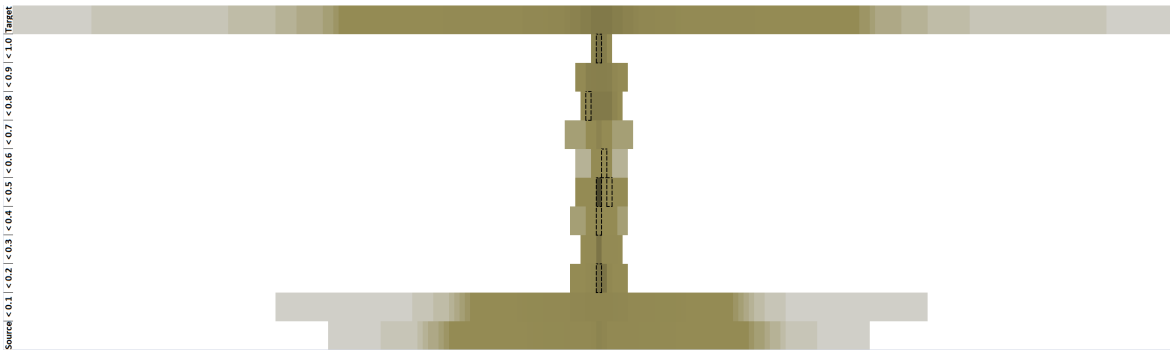


Figure 2.7: A visualization of the Rat metabolic network that places vertices in the vertical direction based on their location metric. Specifically, we discretize the location metric in 12 bins (the lowest bin for sources, the highest for targets, and the 10 intermediate bins for intermediate vertices with each bin accounting for $1/10$ of the $0-1$ range). The path centrality of each vertex is represented by its color (darker for higher path centrality). Vertices with higher centrality are placed closer to the vertical mid-line. The core vertices are represented by dotted rectangles.

sources or low complexity vertices) but where that bias is not strong enough to generate an "ideal hourglass" in which a small set of intermediate vertices is traversed by all ST-paths.

Figure 2.7 is a visualization of the Rat metabolic dependency network that places vertices in the vertical direction based on their location metric (see caption for more details about this visualization). Note that the highest path centrality vertices tend to be at intermediate locations – but some of the sources and targets in this network also have high path centrality. Also, about half of the core vertices are located close to the center of the network (location=0.5), while the rest are closer to sources or targets. The path centrality and the weight of each core vertex are shown in Table B.4. The location of the core vertices varies significantly across different networks. Similar visualizations for the other five networks are given in Figure B.1.

2.4.3 Which are the vertices at the waist of the hourglass?

The complete list of core vertices for each dependency network, together with a short description, the path centrality and the weight of core vertex, are given in the *Appendix*. Here we comment on the qualitative properties of the waist vertices for each network.

The three vertices at the core of the OpenSSH call-graph are shown in Table B.1. They are functions to send and receive network packets, and to execute Unix shell commands. This is not surprising given that OpenSSH is a communication-oriented utility that can be used as a secure remote terminal, among other applications.

The Apache Math library has a core with nine vertices, listed in Table B.2. These methods cover floating point arithmetic operations, matrix decomposition, vector computations, and the "constructors" of some classes related to mathematical and geometric objects.

The vertices at the waist of the two metabolic networks are shown in Tables B.4 and B.6. In biochemistry, the following twelve *precursors* are often considered as the most important metabolites, providing an interface between the different catabolic pathways with the various biosynthesis pathways: Glucose-6-Phosphate, Fructose-6-Phosphate, Glycerone Phosphate, Glyceraldehyde 3-Phosphate, Phosphenol Pyruvate, Pyruvate, Ribose-5-Phosphate, Erythrose-4-Phosphate, Acetyl-CoA, a-ketogluterate, Oxalocetate, and Succinyl-CoA [108, 5, 113]; it is not clear however if these precursors are equally important for every species or if the previous list should include additional metabolites. In the case of Rat metabolic network, the identified waist includes eight of the previous precursors, plus few more key compounds for the synthesis of enzymes, lipids, fatty acids, etc. In the case of the Monkey metabolic network, the waist includes seven precursors. Several waist vertices are the same with those in the Rat (or similar, in the case of SCCs or PES).

The vertices at the waist of the two citation networks are shown in Tables B.8 and B.9.

The Cornell Legal Information Institute (CLII) lists several *landmark* SCotUS cases for every major legal matter in the US [71]. This classification of cases as landmarks is based on input from legal experts. All court cases that appear in the waist of the Abortion network are also listed as landmarks by CLII. In the Pension network, five out of the seven waist vertices are also listed as landmarks by CLII.

2.5 Likelihood of observing the hourglass effect in random networks

We evaluate the statistical significance of the observed hourglass effect in real networks in this section. From given real networks, we create random networks and measure the likelihood of the hourglass effect occurring in them. The randomization process preserves key properties like number of vertices and edges, in-degree of vertices and partial ordering among vertices of the given real network. It however reassigns edges between pairs of vertices and changes the out-degree of vertices. We describe it in detail below.

2.5.1 Randomization of dependency networks

A prior work on randomization of directed acyclic graphs (DAGs) was done by Karrer and Newman [58]. In their method, they preserve the total order among network vertices and in and out degree of each vertex of the original network after randomization. We however, seek to preserve a different set of attributes and propose the following randomization process tailored for that.

For a given dependency network, we preserve the in-degree of each vertex, and the edges of the randomized network do not violate the partial ordering of vertices in the original network. If a vertex u has a path towards vertex v in the original network, then u is an ancestor vertex of v. The set containing all ancestor vertices for v is denoted with $A(v)$.

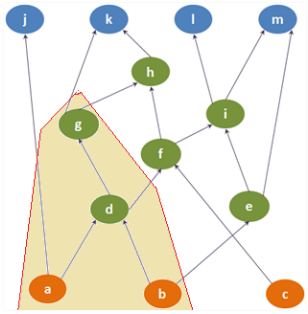


Figure 2.8: The ancestors of vertex g are d, a, b. The in-degree of g is 1. So in the randomized network, vertex g will have one new incoming edge picked randomly from its 3 ancestor vertices.

For a pair of vertices u, v, we can have one of the following (1) u is an ancestor of v, (2) v is ancestor of u, or (3) u, v are unrelated. Preservation of partial ordering among vertices signifies the following two things:

1. if u is an ancestor of v in the original network, then we can not have that v is an ancestor of u in the randomized network.

2. if u and v are unrelated in the original network, they are also unrelated in the randomized network.

For each vertex v in the original network, we remove all incoming edges. We then randomly pick in-degree(v) number of distinct vertices from $A(v)$ and add edges from those vertices to v. This process preserves all the earlier mentioned properties of a given real network. The randomization mechanism is illustrated in Figure 2.8.

2.5.2 Randomization results

For each of the real networks described in Section 2.4, we generate 100 random networks and plot the distribution of the H-score of the random networks to compare with the original network. Figure 2.9 shows these results. The H-score of the random networks is signif-

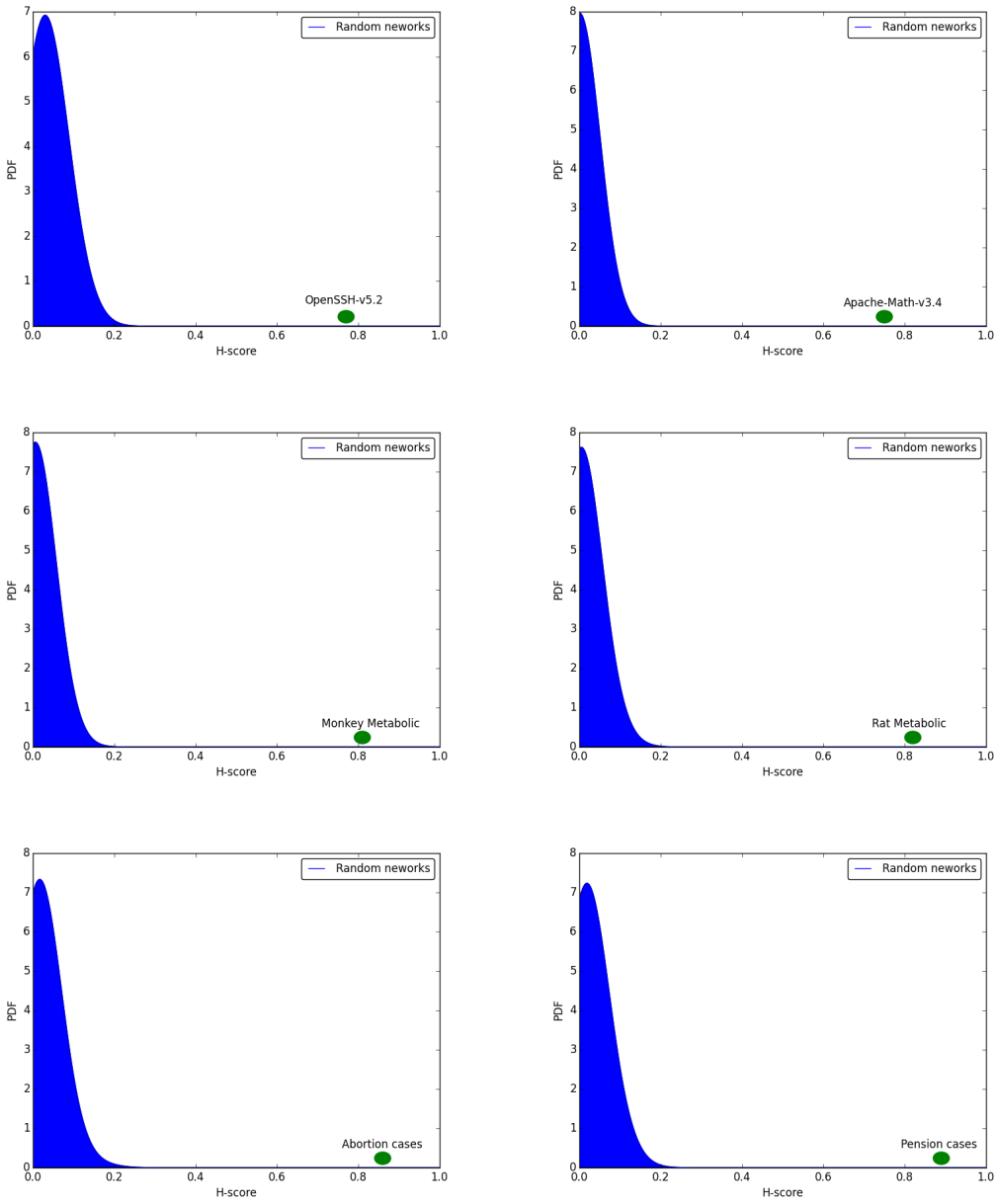


Figure 2.9: Randomization of the six dependency networks we analyzed earlier to evaluate the likelihood of observing the hourglass effect by chance.

icantly less than the corresponding original network. The randomization process largely removes the hourglass effect that was present in these real networks. In Section 3.2, we describe another randomization model that does not decimate the hourglass effect in networks and we discuss further why the randomization method presented in this section is reducing the hourglass effect significantly.

2.6 Comparison with other methods

In this section we compare (1) the path centrality metric of our framework with other centrality metrics, and (2) the core detection method provided by the hourglass framework with other well known network core detection methods. The focus of this section is to show that the metric and framework we developed attempt to extract a different characteristic of the network vertices than these other methods do. We utilize a dataset described earlier: the SCoTUS case citation network for "abortion" and "pension/benefits" related cases, apply the various centrality metrics and core detection methods on these two networks and compare the results across various other methods with our method.

2.6.1 Comparison with existing centrality metrics

We compare our path centrality metric with the following classical centrality metrics from literature:

- **Degree centrality** is a measure of the number of edges incident to a vertex.
- **Closeness centrality** is a measure of the average length of the shortest paths between a vertex with all the other vertices.
- **Betweenness centrality** is a measure of how frequently a vertex lies along the short-

est path(s) between two other vertices.

- **Katz centrality** is a measure of how many other vertices are connected to a vertex with connections to distant vertices penalized.
- **Eigen centrality** is a measure of influence or connection strength of a vertex computed by considering the connection strength of the vertices it is connected to.
- **Pagerank centrality** is a variant of the Katz and Eigen centrality where three distinct factors determine the importance of a vertex: (i) the number of links it receives, (ii) connection propensity of the linkers, and (iii) the centrality of the linkers.

Note that the SCoTUS citation network is a acyclic network. Among the classical centrality metrics, Eigen centrality is not applicable to acyclic networks. To apply Eigen centrality on the SCoTUS network, we ignore directionality. We compute all the centrality metrics using the Python NetworkX library [45].

We start with extracting the top-10 cases based on the path centrality metric. For each of these cases (ranked 1 to 10), we show their corresponding ranks based on all the other centrality metrics for the "abortion" cases in Table 2.3. Here are some key observations from the table:

- Roe v. Wade is widely regarded as one of the most important cases in the abortion legality history. It comes at rank-2 in our metric. It is also a top ranked case according to Degree, Betweenness and Eigen centrality metric. For all the remaining metrics however, it is not a top ranked case.
- Degree and Betweenness centrality has some of the top path centrality ranked cases also as their top ranked cases. Note that betweeness centrality share some similarity with our metric in that it also gives importance to vertices through which a large number of shortest paths between any pair of vertices go through. Since these shortest

Table 2.3: Rank of the top-10 path centrality vertices according to other centrality metrics for the abortion cases.

Top-10 Ranked Path Centrality Cases	Rank in Centrality metric					
	Degree	Closeness	Betweenness	Katz	Eigen	Pagerank
1. Planned Parenthood v. Casey (1991)	6	1	13	1	2	1
2. Roe v. Wade (1973)	1	379	2	35	1	32
3. Harris v. McRae (1980)	9	54	6	13	6	38
4. Webster v. Reproductive Health Services (1988)	10	10	32	3	7	8
5. Doe v. Bolton (1973)	8	385	10	51	3	45
6. Bigelow v. Virginia (1974)	12	276	8	34	28	40
7. VA St. Bd. of Pharm. v. VA Citizens Consum. Coun. (1975)	2	220	3	24	15	35
8. Carey v. Population Services International (1977)	16	117	11	21	5	61
9. Jacobellis v. Ohio (1964)	19	485	9	91	153	37
10. Maher v. Roe (1976)	22	149	20	27	10	67

Table 2.4: Rank of the top-10 path centrality vertices according to other centrality metrics for the pension cases.

Top-10 Ranked Path Centrality Cases	Rank in Centrality metric					
	Degree	Closeness	Betweenness	Katz	Eigen	Pagerank
1. Goldberg v. Kelly (1970)	1	89	1	12	1	15
2. US R. Ret. Bd. v. Fritz (1980)	5	18	2	15	5	28
3. Allied Structural Steel Co. v. Spannaus (1977)	8	74	3	8	243	16
4. Johnson v. Robison (1974)	4	77	4	16	2	18
5. Plaut v. Spendthrift Farm (1994)	6	5	12	3	132	3
6. City of LA Dept. of Water & Power v. Manhart (1977)	14	284	6	50	252	79
7. Nachman Corp. v. Pension Benefit Guaranty Corp (1979)	59	96	5	77	153	100
8. Richardson v. Belchar (1971)	100	154	16	114	3	272
9. San Antonio Indep. Sch. Dist. v. Rodriguez (1972)	100	154	16	114	3	272
10. Intl. Union, United Auto v. Johnson Controls (1990)	26	19	19	18	397	23

paths are a large subset of the paths we consider, we can expect somewhat closer results compared to other centrality metrics. The Degree centrality metric on the other hand is not related to our method in any ways. The reason we see some very close matches and some other missed matches with Degree centrality is because, a vertex with a large Degree centrality is expected to have a large path centrality, however the opposite is not true. We can have vertices with low number of connections, but many paths going through them (consider the ideal hourglass network example).

A similar comparison for the "pension" network is presented in Table 2.4. We again see a closer match with the Degree and Betweenness centrality. Some of the top ranked path centrality cases, are of much lower ranks in Degree centrality. The deviation Degree centrality has in its ranks on this network is much larger corroborating our earlier point.

Table 2.5: Jaccard Similarity of various centrality with path centrality for the top-20,50 vertices.

Centrality	Dataset			
	Abortion cases		Pension cases	
	top-20	top-50	top-20	top-50
Degree	0.48	0.49	0.29	0.28
Closeness	0.11	0.16	0.17	0.36
Betweenness	0.42	0.61	0.60	0.47
Katz	0.25	0.42	0.25	0.31
Eigen	0.48	0.40	0.17	0.07
Pagerank	0.14	0.36	0.21	0.38

We computed the Jaccard Similarity (which compares how similar two sets are and assigns a score between $[0-1]$ with 0 being completely dissimilar and 1 being exactly same) between the top-20/50 cases based on path centrality and other centrality metrics. Betweenness centrality consistently produces results having largest overlap with our results (around 50% match). Degree and Eigen centrality show very different result for the two datasets. For the abortion cases, they show relatively large similarity, where as for the pension cases, they shows the complete opposite result. Eigen centrality is similar to Degree centrality but differs in that it matter where connections are coming from. But in the same way Degree centrality does not capture the dependency path based importance, the Eigen centrality also fails to isolate low connection, high path centrality vertices. The remaining centrality metrics provide consistently different result from ours. The Closeness and Katz centrality find important vertices by considering how topologically central (i.e. close in terms of hop distance) they are to all other vertices. This again is a very different characteristic than what we seek to isolate in a dependency network with path centrality. Consider an hourglass network where the waist of hourglass is not at the physical middle, but near to either the sources or the targets. Neither Closeness or Katz centrality can correctly identify the importance of the waist vertices solely based on their topological position. Lastly, the Pagerank centrality is a specialized form of the Eigen cenrality and shares same differences with our method.

Table 2.6: Kendall's τ correlation with path centrality. p-value for all entries < 0.001.

	Dataset	
Centrality	Abortion cases	Pension cases
Degree	0.45	0.36
Closeness	0.30	0.43
Betweenness	0.33	0.32
Katz	0.29	0.41
Eigen	0.45	0.40
Pagerank	0.28	0.40

In Table 2.6 we compute the Kendall's-τ rank correlation between path centrality with other centralities across all vertices. We see variability in scores among the other centralities for the abortion cases where Degree and Eigen centrality show better correlation. For the pension cases however we see a similar correlation across all the other centrality metrics. The observed correlation is not strong between any of the other centrality metrics with path centrally providing us the insight that path centrality indeed captures a very different aspect of the vertices in a dependency network than these other classical centrality metric do.

Classical centrality metrics were not developed to evaluate vertex importance in source-target dependency networks in particular. Hence the generalized characteristics that they evaluate on do not address the unique features of a source-target dependency network. The notion that dependencies run from source to targets and some vertices behave as the backbone by shouldering a large number of these dependencies is not addressed by these other metrics. The path centrality metric on the other hand is a specialized metric tailored to work on the source-target dependency networks and will identify those key dependency backbone vertices. Consequently we do not expect it to perform equally well in a generalized network settings.

2.6.2 Comparison with existing network "core finding" algorithms

We compare the core vertices identified by our hourglass framework with the cores extracted by the following well known "core finding" algorithms:

- **Core-periphery algorithm.** Core-periphery analysis partitions a network into two discrete classes: a densely connected subgraph forms the the core and the remaining network vertices that are sparsely connected to both the core and among themselves forms the periphery.

- **k-Core algorithm.** A k-Core is a largest subgraph of a network where the vertices have k or more connections.

- **Rich-club algorithm.** Rich-clubs are high degree network vertices that mostly form a clique among themselves. Formally, The rich-club coefficient for a given network N is then defined as:
  $\phi(k) = \frac{2E_{>k}}{N_{>k}(N_{>k}-1)}$
  where $E_{>k}$ is the number of edges between the vertices of degree greater than or equal to k, and $N_{>k}$ is the number of vertices with degree greater than or equal to k. Since higher-degree vertices have higher probability of sharing connections with each other, even random networks can have increasing rich-club coefficients as k is increased. To control for this effect, the rich-club coefficient is normalized relative to the rich-club coefficients of comparable random networks as follows:
  $\phi_{norm}(k) = \frac{\phi(k)}{\phi_{rand}(k)}$
  where $\phi_{rand}(k)$ is the average value of ϕ_k across many random networks. The existence of rich-club organization is defined by $\phi_{norm}(k) > 1$ over some range of values of threshold degree k.

The core finding algorithms mentioned are developed for undirected networks. The un-

Table 2.7: Extracted core size of various core detection algorithm.

Core detection algorithm	Dataset	
	Abortion cases	Pension cases
Core-periphery	39 (3%)	30 (2%)
k-Core	25 (2%)	28 (2%)
Rich-club	-	-
Hourglass-waist	4 (0.2%)	11 (0.8%)

Table 2.8: Jaccard Similarity between the cores extracted by various core detection algorithm and the hourglass core.

Core detection algorithm	Dataset	
	Abortion cases	Pension cases
Core-periphery	0.10	0.33
k-Core	0.12	0.08
Rich-club	0.14	0.47

derlying principle for most of them is based on considering connections among vertices, without considering directionality. Hence we apply all the aforementioned algorithm on the undirected case networks. For the hourglass core finding algorithm, we of course require the network to be directed. For the core-periphery algorithm we use the Borgatti-Everett method first introduced by Borgatti and Everett [15] and implemented by Kojaku and Masuda [67]. For k-Core and rich-club, we use the Python NetworkX implementations. The core for the hourglass framework was extracted for $\tau = 90\%$.

Table 2.7 shows the size of the cores extracted by various methods. The core-periphery model and k-Core model extracts similar sized cores containing about 2%-3% of all network vertices. Unfortunately, there were no rich-clubs observed in either of the case networks based on the normalized rich-club coefficient. Instead, we used a set of highest degree vertices as proxy rich-club in subsequent analysis. The hourglass core was much smaller comparatively containing $< 1\%$ of the network vertices. In Table 2.8, we compare the Jaccard Similarity of the core extracted by hourglass framework with the other cores. In this table, the rich-club vertex set had cardinality equaling corresponding hourglass core size. The core for both the networks from the other methods have weak similarity with our

Table 2.9: Precision and recall of the cores extracted by various core detection algorithm based on the landmark cases. For the hourglass core, we show scores for two extended cases. The regular one it for τ=0.9. In extension a., we modify τ so that the hourglass core size is equal to the number of landmark cases for the two networks. In extension b., we modify τ so that the hourglass core size matches to core-periphery model's core.

Core detection algorithm	Dataset			
	Abortion cases		Pension cases	
	Precision	Recall	Precision	Recall
Core-periphery	0.28	0.73	0.17	1.00
k-Core	0.40	0.67	0.04	0.20
Rich-club	0.53	0.53	0.20	0.20
Hourglass core (τ=95%)	1.00	0.27	0.36	0.80
Hourglass core Ext. a	0.47	0.47	0.60	0.60
Hourglass core Ext. b	0.33	0.87	0.17	1.00

core specially for the abortion network. We observed that in most cases hourglass core is a small subset of the other larger (core-periphery and k-Core) cores.

For the SCoTUS case networks, we have a set of cases that were identified as influential and landmark across various categories by the Legal Institute of Cornell Law School [71]. These landmark cases are selected based on knowledge of domain experts and are not related to any network analysis method. In Table 2.9 we compute the quality of cores from the classical core finding methods and hourglass core against the landmark cases for both networks using precision and recall. Precision is the fraction of relevant instances (i.e. the landmark cases) among the retrieved instances (i.e. the extracted cores), while recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. In this table, the rich-club vertex set had cardinality equaling corresponding number of landmark cases. In the abortion network, there were a large number (15) of landmark cases. The hourglass core had an excellent precision but poor recall due to retrieval of a small core. The other methods retrieved large cores and had low precision and good recall. To get a fair comparison with the other methods, we increased the τ parameter of our core detection framework to match (1) the number of the landmark cases, and (2) the core size of the other methods. For the first case, we get a moderate precision and recall, and for the second case we get scores equivalent or better than the other cases. For the pension

cases (which had 5 landmark rulings), we see a different result. The original hourglass core performs much better than the other methods in precision and performs equivalently for the best recall score. When we increased the hourglass core size to match core-periphery core, its precision and recall equalled the core-periphery case. However, the precision dipped a lot due to addition of a lot of false positives. The k-Core algorithm had a poor performance in particular for the pension network. The proxy rich-club had a strong performance for the abortion network, but a poor performance for the pension network. For both networks, the precision of the original hourglass core was much better than the other methods signifying the strong quality of hourglass core.

A key aspect of the hourglass framework is, it gives us both the ability to find importance of individual vertices as well as extracting a set of core vertices with tunable cardinality. On the other hand core-periphery, k-Core and rich-club framework's outcome gives a binary separation of central and non-central vertices. The core of these other methods are mostly determined based on their connectivity, not based on their role in supporting source-target dependencies. In short these other methods try to answer different questions not particularly pertinent for dependency networks. Hourglass framework on the other hand examines if the vertices form a hierarchy by considering the source to target paths, in which a small set of vertices at intermediate layers of the hierarchy cover most of those paths. Hence we believe the core detection method of the hourglass framework can extract vertices that play important role is supporting the source-target dependency flows.

2.7 Summary

In this chapter we introduced the concept of source-target dependency paths in general, non-layered dependency networks. We defined the path centrality metric based on source-target paths to quantify the topological importance of dependency network vertices. We

argued that path centrality metric is more appropriate for identifying important vertices in a dependency network than other classical centrality metrics. To identify the most central modules of the underlying system, we formulated the τ-core problem, and provided a heuristic solution for the problem to extract the smallest subset of core vertices through which at least $\tau\%$ of all source-target paths traverse.

To quantify the hourglass effect in dependency networks, we introduced the H-score that computes the ratio of the core size in a dependency network and its corresponding flat (i.e., non-hierarchical) network that preserves the source dependencies of each target in the original network. We also defined a location metric for each vertex to capture its relative position in the dependency network between sources and targets and utilize it for visualizing the hierarchy and core positions in the network.

We applied the hourglass analysis framework in six dependency networks from three different disciplines: two call-graphs (software engineering), two metabolic networks (biology, biochemistry) and two citation networks (information science). All the networks exhibited the hourglass property but to a varying degree and with different waist characteristics. For some of the networks, we had available a set of well known important modules (extracted by domain experts) and we found that they had a large overlap with the corresponding hourglass core.

We measured the likelihood of observing the hourglass effect simply by chance in the real dependency networks by creating comparable random networks. We found that the randomization process that preserves the number of source-target-intermediate vertices, in-degree and partial ordering of each vertex but only perturbs the dependency relationship of vertex do not preserve the hourglass property of the real networks.

Finally we did a quantitative analysis using real networks between the hourglass framework and existing network core finding methods like core-periphery structure, k-core and

rich-club. We also compared the path centrality metric with existing network centrality metrics. These studies revealed that, the hourglass framework identifies a different set of vertices as more important than these other methods and is inherently evaluating a different network aspect pertinent to dependency networks than these other methods.

CHAPTER 3

WHAT CAUSES THE HOURGLASS EFFECT

3.1 A model of dependency network formation

What determines whether a dependency network will exhibit the hourglass property or not? Let us think about this question in the context of Lego-like toys, in which a vertex v corresponds to a Lego module and its incoming edges show which simpler Lego modules are required to put v together. The sources correspond to the given elementary building blocks and the targets correspond to the final objects we want to construct. One extreme approach is to create every object only from the elementary blocks, without reusing any intermediate modules that have been previously constructed. Another approach is to reuse as much as possible intermediate modules, expecting that this will require less work. In practice, of course, the design approach is always somewhere in the middle, with more complex intermediate modules constructed from simpler intermediate modules as well as elementary blocks.

To understand the implications of this "preference for reuse", we present here a simple, probabilistic model for the gradual formation of a dependency network. The model focuses on how each new vertex selects its incoming edges among the set of vertices that have been previously constructed. Through a single *reuse parameter* α, the model generates dependency networks in which every new vertex depends on either mostly sources (leading to flat, non-hourglass networks) or on the more recently constructed intermediate vertices (resulting in hourglass networks), or anything in between.

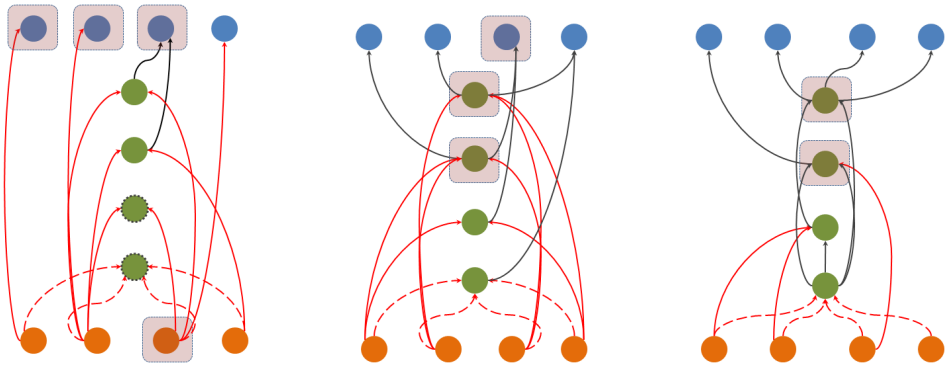
We refer to the following model as *Reuse-Preference* or *RP-model*. There are V vertices that consist of S sources, M intermediates and T targets. The vertices arrive in the network, or they are created, sequentially or in batches, as follows. First, all sources are created at the same time; they represent the elementary modules of the underlying system. Then, the intermediate vertices are created sequentially (the case of batch arrivals is considered in Section 3.2). Suppose that v is the m'th intermediate vertex that has arrived in the network, with $1 \leq m \leq M$. We assign vertex v to *rank-0*, and the previously created $m-1$ intermediate vertices to *rank-1* through *rank-(m-1)* (in order of arrival – the oldest intermediate vertex always has *rank-(m-1)*). The S sources are randomly given ranks m through $m+(S-1)$. Note that the ranking changes every time a new vertex is added. The T targets are created in a batch at the end of the network formation process, and they are given the same rank (*rank-0*).

Suppose that we are given the in-degree $d_{in}(v)$ of v. The origin of every incoming edge to v is determined as follows. When the m'th intermediate vertex v is created, we select the vertices it will depend on probabilistically. In the following, we use the Zipf distribution (but other statistical models could also be used). Specifically, the probability that v will have an incoming edge from a vertex u at *rank-r* is given by:

$$\text{Prob}[(u,v) \in \mathbf{E}] = \frac{r^{-\alpha}}{\sum_{i=1}^{S+m-1} i^{-\alpha}}, \quad 1 \leq r \leq S+m-1 \tag{3.1}$$

The incoming edges to the T target vertices are determined in the same way; note that a target will never by connected to another target because all targets are added in the same batch, having *rank-0*. Additionally, we artificially exclude the possibility of multi-edges.

When $\alpha = 0$ the newly created vertex v selects dependencies uniformly across all earlier vertices. As α increases above zero, v has a preference for more recently constructed vertices, increasing the level of reuse in the dependency network. On the other hand, as

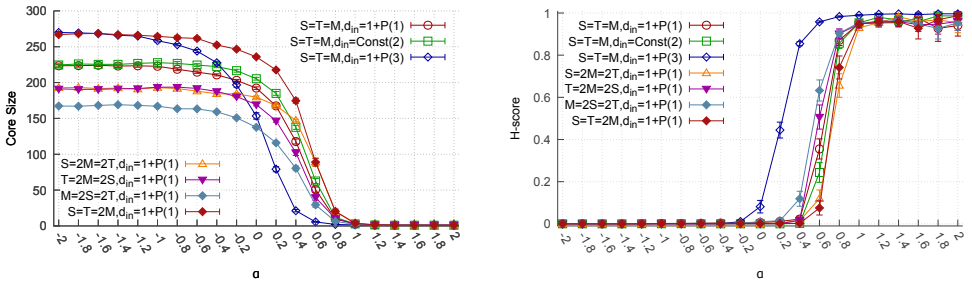


(a) α=-1, H-score=0.0 (b) α=0, H-score=0.25 (c) α=+1, H-score=0.5

Figure 3.1: Three dependency networks generated by the RP-model for different values of α (V=12, S=T=M=4, d_{in}=1 + Poisson(1), α=$\{-1, 0, 1\}$, and τ=0.90. The sources are shown in orange, the targets in blue, and the intermediates in green. Vertices that do not belong to any ST-path are shown as dotted. The edges from all sources to the first intermediate vertex v, shown with red dashed arrows, have unit weights. The weight of edges from sources to other intermediate and target vertices, shown with red-solid arrows, is increased to $S/d_{in}(v)$. The remaining edges, shown with solid black arrows, have unit weights. The core vertices for each network are shown in boxes.

α decreases below zero, v has a preference for older vertices, i.e., closer to the sources, decreasing the level of reuse.

For large values of α, it is possible that many sources will not be chosen by any vertex higher in the hierarchy. To ensure that there are no disconnected sources (i.e., elementary blocks that are not utilized by any other module), we add an edge from every source to the first intermediate vertex, say v. So, instead of its originally assigned in-degree $d_{in}(v)$, vertex v now has S incoming edges. These extra edges however can inflate the path centrality of v and of any vertices that depend on v. To maintain the path centrality of v relative to the rest of the intermediate and target vertices, we need to increase the weight of the edges from sources to other vertices by a factor $S/d_{in}(v)$. To avoid fractional weights, the weight of the extra edges from sources to v is set to 1, the weight of the original edges from sources to other vertices is set to $S/d_{in}(v)$, and $d_{in}(v)$ is sampled so that the previous ratio is an integer. The rest of the edges have a weight of 1.

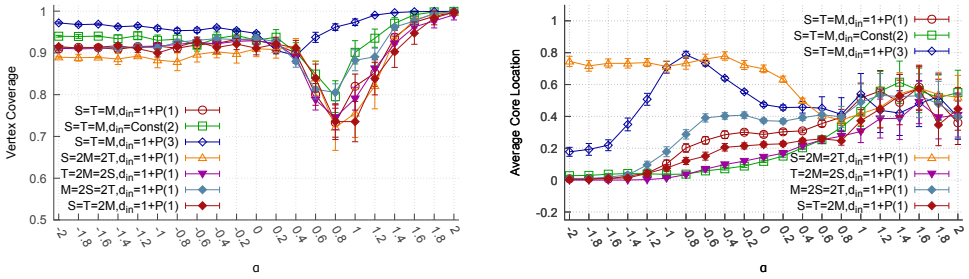


(a) Core size (b) H-score

Figure 3.2: Effect of α on the core size and H-score metric.

Figure 3.1 shows three small dependency networks constructed using the RP-model for three different values of α. When $\alpha = -1$, almost all ST-paths are directly connecting sources to targets (little reuse of intermediate vertices), and most intermediate vertices are not used in the construction of any target (shown as dotted). The core consists of a combination of sources and targets, and it is relatively large (in this example, equal to the number of sources or targets). On the other hand, when $\alpha = +1$, the preference to connect to higher complexity vertices leads to longer dependency paths. A small number of intermediate vertices are traversed by a large fraction of ST-paths, just based on chance, and so those vertices end up with much higher path centrality than most other vertices. The core of such dependency networks is then small, relative to the number of sources or targets, and those networks have high H-score.

In the following, we illustrate the behavior of the RP-model with computational experiments. All networks have V=1000 vertices but we vary the proportion of sources, targets and intermediate vertices. The path coverage threshold τ is set to 90%, unless stated otherwise. The in-degree of each vertex is either constant (denoted as "d_{in}=Const(x)") or set to $1 + \text{Poisson}(x)$ where x is the mean of a Poisson distribution (denoted as "d_{in}=1+P(x)"). All results are based on 100 simulation runs, and they are reported with 95% confidence intervals.



(a) Core vertex coverage

(b) Average core location

Figure 3.3: Effect of α on the core vertex coverage and average core location metrics.

Figures 3.2 and 3.3 show the effect of the reuse parameter α on the core size $C(\tau)$, the H-score $H(\tau)$, the core vertex coverage $U_{\mathbf{C}}$, and the average core location $L_{\mathbf{C}}$. Each graph shows results for seven sets of network parameters, varying the proportion of sources, targets, intermediates, and the in-degree values and distribution. For example, the label $S = 2M = 2T$ means that S=500 and M=T=250 (so that V=1000).

Let us first focus on negative values of α:

a) As α decreases below zero, it becomes more likely that targets connect directly to sources (see the "direct" network of Figure 1.1-b or Figure 3.1-a). Most intermediate vertices are not included in any ST-path, their path centrality is close to zero, and so they are not included in the core. Instead, the core consists of mostly a combination of sources and targets. To cover the large fraction τ (90%) of these direct ST-paths however, the core needs to include many vertices. For instance, in the scenario M=2S=2T the core has about 160 vertices, while $\min\{S, T\}$=250. The higher the average in-degree is, the larger the core needs to be (to cover the increased number of ST-paths).

b) The corresponding flat dependency network is similar to the original network in terms of how sources and targets are directly connected, and so it has approximately the same core size; this is why the H-score is close to zero.

c) The core vertex coverage is close to one for the following reason: if all ST-paths are direct connections between sources and targets and the core covers a fraction τ of these paths, the core vertex coverage will be at least $1 - 2(1 - \tau)$ because every non-covered ST-path contributes at most two non-covered vertices.

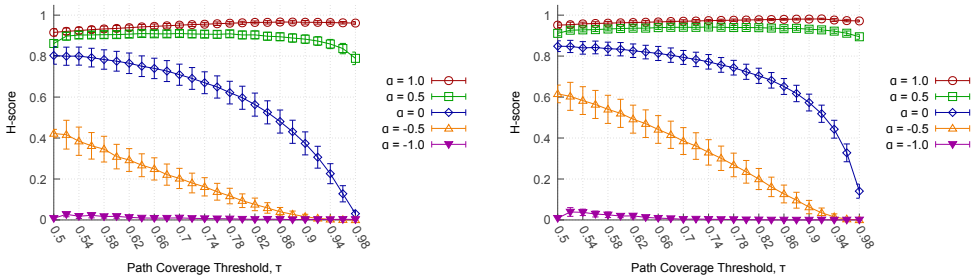
d) The location of the core varies significantly with the network parameters because the core consists of mostly sources and targets. So, if the core consists mostly of sources (as in the T=2M=2S scenario) the core location moves closer to zero, while if the core includes mostly targets (as in the S=2M=2T scenario) the core location moves closer to one.

Let us now focus on positive values of α:

a) As α increases above zero, each target or intermediate vertex prefers to connect to vertices that are close to it in the given hierarchy (see Figure 3.1-c). So, the ST-paths become longer and some intermediate vertices get to be traversed by a larger fraction of ST-paths (just based on chance). Vertices with high path centrality tend to form the core of the dependency network, and their number gradually drops as α increases.

b) The core of the flat network, on the other hand, is much larger, as in the case of negative α, and so the corresponding H-score approaches its maximum value (one) as α increases. The transition point, from $H(\tau) \approx 0$ to $H(\tau) \approx 1$, shifts towards lower values of α as the density of the network increases (see scenario d_{in}=1+P(3)) because the likelihood that few intermediate vertices will acquire much higher path centrality increases.

c) The core vertex coverage curves follow an interesting pattern: as α increases from negative values to positive values, $U_{\mathbf{C}}$ first decreases and then increases. During the transition



(a) $d_{in} = 1 + \text{Poisson}(2)$ (b) $d_{in} = 1 + \text{Poisson}(3)$

Figure 3.4: Effect of path coverage threshold τ on H-score, for different values of α. Network parameters: S=T=200 and M=600 (V=1000).

from a flat network ($H(\tau) \approx 0$) to an hourglass-like network ($H(\tau) \approx 1$), it is common for ST-paths to traverse one or more intermediate vertices that are not traversed by many other ST-paths (see the "decoupled" network of Figure 1.1-c). So, in that transition range, the fraction $1 - \tau$ of ST-paths that are not covered by the core account for more than $2(1 - \tau)$ non-covered vertices (because they include one or more intermediate vertices). As α further increases, the core is traversed by an increasing fraction of ST-paths, eventually covering almost all ST-paths, and so also covering almost all vertices that appear in ST-paths.

d) The location of the hourglass waist is gradually converging towards the middle of the dependency network, i.e., $L_{\mathbf{C}} \approx 0.5$. We should note that the location of a PES is, by definition, equal to the median location of the vertices in that set. So, one reason that the location of the waist converges to 0.5 as α increases is that the waist in that regime often includes a large PES with many intermediate vertices that have locations between 0 and 1.

Finally, Figure 3.4 shows the effect of the path coverage threshold τ on the H-score for few different values of α. When the reuse parameter α is close to one (or higher), the H-score is almost one, largely independent of τ, meaning that the hourglass property is robustly established.[1] When α is negative or even close to zero, on the other hand, the

[1]The slight increase of the H-score with τ, when α=1, is because the core size of the flat network increases faster than the core size of the original network, as τ increases.

H-score is typically less than 50% and so those networks clearly do not have the hourglass property, independent of the selection of τ. For intermediate values of α, the H-score depends on the selection of τ and on other network parameters, such as the average in-degree.

3.2 Fitting the RP-model to a given dependency network

We now describe how to parameterize the RP-model so that it produces random networks G that have approximately the same H-score with a given dependency network G'. We also compare these synthetic networks with G' in terms of the path centrality distribution, the out-degree distribution, and some more network metrics that are relevant to dependency networks.

Given G', we can easily identify its set of sources and targets. A synthetic network G will have the same set of sources, targets, and intermediate vertices. Since it may not be possible to identify a global ordering between the intermediate vertices, we place the vertices of G in layers based on the topological sorting of G', as follows. First, all sources of G' are placed at layer-0 of G. Then, recursively, we place at layer i of G those intermediate vertices of G' that depend on at least one vertex of layer $i-1$ (for $i > 0$). Finally, the targets of G' are placed at the top layer of G (independent of the layer of their incoming edges). This layered representation of G gives a partial ordering relation between vertices: the vertices of layer i are supposed to arrive (or to be created) as a batch, and they do not depend on each other.

The in-degree $d_{in}(v)$ of each non-source vertex v in G is the same with G'. To generate the specific inputs of v, we identify the set of ancestors $A(v)$ of v in $G' - v$ depends directly or indirectly on these vertices. When a vertex v is created at layer i, it can receive incoming

Table 3.1: Fitting the RP-model to the six dependency networks of Section 2.4: we show the α estimate, the average ST-path length, the core size (for the same value of τ as in the analysis of the original networks – see Table 2), the core vertex coverage $U_{\mathbf{C}}$, and the average core location $L_{\mathbf{C}}$. The corresponding values for the original dependency networks are shown in parentheses.

	Networks					
	Software Call-graphs		Metabolic Nets		SCotUS Citation Nets	
	OpenSSH v-5.2	*Apache Math v-3.4*	*Rat*	*Monkey*	*Abortion Cases*	*Pension Cases*
α estimate	1.1	2.3	2.4	2.5	2.7	2.3
Core size	3 ± 1 (3)	4 ± 1 (9)	4 ± 4 (7)	4 ± 2 (8)	5 ± 0.5 (4)	8 ± 1 (11)
H-score	0.69 ± 0.05 (0.77)	0.78 ± 0.03 (0.75)	0.76 ± 0.07 (0.82)	0.75 ± 0.04 (0.81)	0.78 ± 0.02 (0.86)	0.87 ± 0.01 (0.89)
ST-path length	8.8 ± 0.8 (10.4)	9.5 ± 0.4 (8.8)	11.3 ± 3.5 (8.3)	12.3 ± 3.8 (8.1)	14.3 ± 0.5 (14.1)	6.4 ± 0.3 (5.1)
Core vertex coverage	0.28 ± 0.04 (0.35)	0.12 ± 0.01 (0.21)	0.32 ± 0.07 (0.53)	0.3 ± 0.07 (0.57)	0.85 ± 0.1 (0.82)	0.45 ± 0.05 (0.48)
Average core location	0.51 ± 0.3 (0.50)	0.05 ± 0.02 (0.12)	0.36 ± 0.12 (0.45)	0.3 ± 0.12 (0.44)	0.2 ± 0.15 (0.74)	0.29 ± 0.18 (0.24)

edges only from vertices in $A(v)$. The selection of inputs of v among the vertices in $A(v)$ is performed probabilistically based on Equation 3.1. [2]

The only difference with the original RP-model is that vertices of $A(v)$ that belong to the same layer have the same rank, and so the same probability of being connected to v.[3]

To parameterize the RP-model, we estimate the value of the reuse preference exponent α so that the synthetic networks G have an H-score that is approximately the same with that of G'. To do so, we generate 100 synthetic networks G for each value of α and compute the average H-score of that sample – the optimal value of α is the value that gives the minimum difference from the H-score of G'.

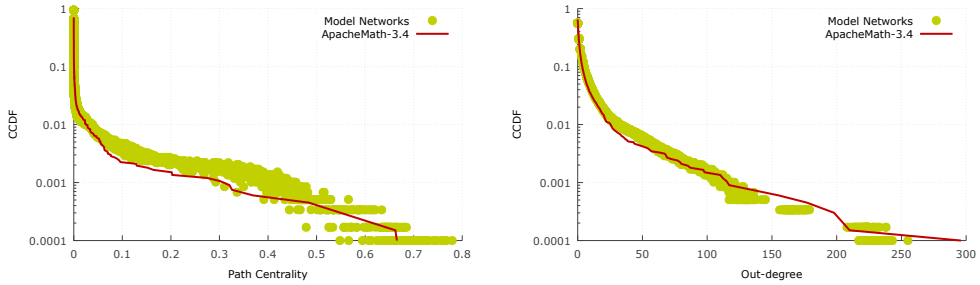
[2]Note the similarities and differences with the randomization method described in Section 2.5. In that randomization method, we start with an hourglass network and randomization significantly removes the hourglass effect. Similar to that method, for a vertex v, input edges are chosen from its ancestors $A(v)$ here as well. In that method however, inputs were selected uniformly randomly from $A(v)$. Here the inputs of vertex v is chosen from $A(v)$ in a non-uniform manner based on Equation 3.1. The value of α is chosen so that the H-score of the random network is same as the original network.

[3]In the original RP-model, vertices are created sequentially and so each layer (other than the boundary layers of sources and targets) includes only one vertex.

Table 3.1 shows the estimate of α for each dependency network of Section 2.4. The average H-score of those synthetic networks is within 10% of the H-score of G'. Note that the value of α in all six networks is much larger than zero (in the range of $[1.1 - 2.7]$). When α is positive, a vertex v with rank $r(v)$ has high probability of getting an input from a vertex u at a rank $r(u)$ where $r(u) < r(v)$ and the rank difference $(r(v) - r(u))$ is very small. Recall that the randomization method presented in Section 2.5 selects the inputs of a vertex v uniformly randomly from $A(v)$ corresponding to the case where $\alpha = 0$. This difference in the value of α illustrates the key property behind the hourglass effect: input selections of a vertex need to have a strong bias to minimize the rank differences instead of being uniformly random.

The α parameter utilized in this section for various networks ranges from $[1.1-2.7]$. Note that for the randomization mechanism described in Section 2.5, the corresponding α value would have been 0, where we uniformly randomly chose inputs from ancestors resulting in significant decimation of the existing strong hourglass effect. This signifies that, if the key property of "bias in input dependency selection towards vertices of similar complexity (or most recently added as in this section)" is maintained, we will observe the hourglass effect even when we randomize actual connections between vertices of the network.

The RP-model generates ST-paths with a similar average length as in the given dependency networks. The average length of the dependency paths is an important metric, as it represents the typical number of intermediate vertices between a source and a target. The synthetic networks are often similar with the given dependency networks in terms of the core vertex coverage and the average core location but there are also some significant deviations (the model overestimates the core vertex coverage of the call graphs and metabolic networks, and it does not predict correctly the location of the core of the SCOTUS abortion cases network).



(a) Path centrality distribution (b) Out-degree distribution

Figure 3.5: Comparing path centrality and out degree distribution of a real network with model generated synthetic networks.

Figure 3.5 shows the path centrality and the out-degree distributions for the Apache Math call-graph and for an ensemble of 100 synthetically generated networks by the RP-model, as described earlier. Similar results for the five other dependency networks are shown at the Supplementary Material (see Figure B.2). Even though there is significant variability between members of the ensemble (both distributions are highly skewed), the model is able to generate distributions of path centrality and out-degree that encompass the main mass of the empirical distributions of G'.

3.3 Comparison with another generative dependency network model

We are not aware of any other model that can generate general non-layered hourglass dependency networks. However, there is a well-known class of models that can generate growing dependency networks based on variations of the "edge-copying" mechanism [66, 68]. The simplest instance of the edge-copying model is: a new vertex v depends with probability β on a randomly chosen vertex u, and with probability $1-\beta$ on a randomly chosen vertex w that u depends on, i.e., v copies an incoming edge of u [66]. If these dependencies are represented with directed edges from u (or w) to v, the out-degree distribution follows a power-law with exponent $-\frac{2-\beta}{1-\beta}$ [69]. For $\beta < 1/2$, the edge-copying model generates scale-free networks and some some vertices are expected to be hubs. An important question

is: can the edge-copying model generate hourglass dependency networks, at least for some values of β? And if so, is it that the hubs appear at the waist of the hourglass network?

We follow the same process as in Section 3.2 to fit the edge-copying model in a given dependency network, i.e., the number of sources, targets and intermediate vertices, the (partial) ordering with which the vertices are created, and the in-degree of each vertex are as in the given dependency network. One special case that we need to address is: what if a vertex v selects to copy (with probability $1 - \beta$) an incoming edge of a source u? Since sources do not have incoming edges, we assume that v should receive an incoming edge from u instead. Also, we do not allow multi-edges.

Figure 3.6 shows the results of fitting the edge-copying model in the OpenSSH call-graph, Rat metabolic network and Abortion cases citation network: the y-axis shows the H-score (average and 95% confidence interval) of 100 synthetic networks generated for different values of the parameter β. Note that the H-score is close to zero throughout the range of β, meaning that the edge-copying model is *not* able to generate hourglass networks. As β approaches one, each new vertex depends on randomly chosen existing vertices – which is also what happens in the RP-model when $\alpha = 0$; we have already seen that such networks do not exhibit the hourglass effect. As β approaches zero, the edge-copying mechanism is applied more often and this causes the emergence of hubs. These hubs, however, tend to be sources because the latter are created first, and so their number of outgoing edges increases faster than other vertices [10]. As a result, most targets are connected directly to sources generating dependency networks with very short ST-paths, a large fraction of disconnected intermediate vertices, and a core that consists of almost all source vertices – consequently, the H-score of such networks is close to zero.

Comparing the RP-model with the edge-copying model, we note that the former is able to generate hourglass networks, when α is close to one or higher, because the preference

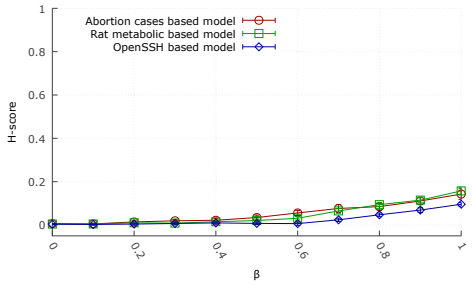


Figure 3.6: H-score of synthetic networks generated by the edge-copying model. The network parameters (number of sources, targets, partial ordering of vertices, and in-degree of each vertex) are set based on the three empirical dependency networks shown in the legend.

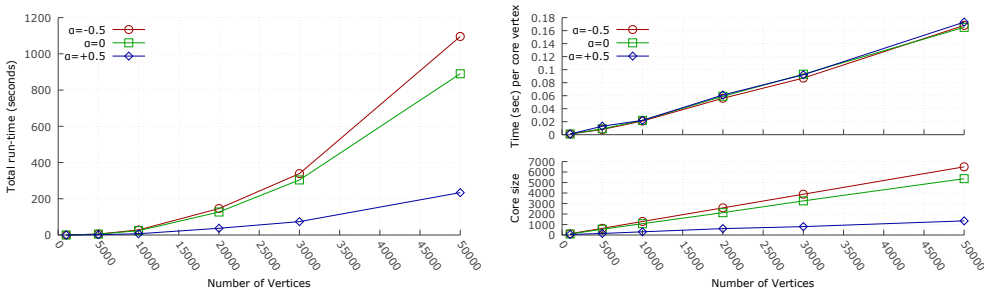
to connect to higher complexity vertices leads to longer dependency paths, and thus to the emergence of few intermediate vertices with much higher path centrality.

3.4 Run-time analysis of core identification algorithm

We can also use the RP-model to examine the scalability of the core identification algorithm. We created synthetic dependency networks of different sizes, for three different values of α (-0.5, 0, 0.5). The proportion of sources and targets remains constant (25% each), while the in-degree of each non-source vertex is $1 + \text{Poisson}(2)$.

As discussed in Section 2.2, the core identification greedy algorithm has a run-time complexity of $O(k\,E)$, where k is the size of the core and E is the number of network edges. In the dependency networks we construct, E increases proportionally with the number of vertices N, i.e., $E = d_{in}\,(N - S)$, where d_{in} is the average in-degree for non-source vertices and S is the number of sources. The relation between k and N is not something we could derive analytically, and it certainly depends on α and the path coverage threshold τ.

Figure 3.7 shows the run-time, the run-time per core vertex, and the core size k as a function of N, for τ=0.90. Note that k increases almost linearly with N for all values of α



(a) Run-time (b) Core-size and run-time per core vertex

Figure 3.7: Run-time analysis of the core identification algorithm using networks generated with the RP-model. The run-time increases quadratically with the network size N. The experiments were run on an Intel-2.5GHz dual-core processor with 6GB of memory.

we consider. Consequently, the total run-time becomes the product of two linear functions of N, and so it *increases quadratically with the network size.* As expected, non-hourglass networks (e.g., when $\alpha = -0.5$) have a larger core, and so they require more computation than hourglass networks.

3.5 Summary

In this chapter, we proposed the "Reuse Preference" (RP) model that captures the bias of new modules to reuse intermediate modules of similar complexity instead of connecting directly to sources or low-complexity modules as a possible explanation for the hourglass effect. We then discussed a framework to parameterize the RP-model so that it produces random networks that have approximately the same H-score with a given dependency network and applied on the real network dataset. The model was able to replicate some key properties such as path centrality distribution, out-degree distribution, core size etc. of the candidate real networks. Finally we used a well known dependency network generator model to compare with the RP model and found that it can not be tuned to form hourglass networks.

CHAPTER 4

THE HOURGLASS EFFECT IN NETWORKS WITH CYCLES: THE *C. ELEGANS* CASE-STUDY

4.1 The *C. elegans* brain network

Caenorhabditis elegans (roundworm) is a nematode that has a relatively simple nervous system that has been extensively and almost completely mapped. Analysis of this system provides us great insight about how the organism gathers external information through its sensory neurons, integrates the information through the interneurons and finally produces relevant response through the motor neurons. With the complete wiring of the nervous system (also called connectome) available, many studies on the connectome based on complex network analysis were done [24, 53, 87, 109, 115, 119]. These studies aim to unveil various interesting properties of the network, including mesoscale organization, topological clustering, wiring cost optimization, central neurons identification, small-world properties etc.

Our focus in this study is on the central neurons of the *C. elegans* connectome treating the network as a hierarchical dependency network. Networks having a well established underlying dependency relationship and strong hierarchy among vertices have been analyzed with hourglass framework previously [100]. Networks having hourglass structure have been shown to have the special characteristic that many input modules are processed and converted to a few universal intermediate building modules which are then reused to create/drive a wide array of output modules. Many real-world networks (e.g. metabolic

networks, software call-graphs, citation network, internet protocol stack) were shown to exhibit the hourglass structure in our earlier works [3, 100]. Having an hourglass structure indicates presence of few central vertices (called the waist of the hourglass or core of the network), that are critical for the network operation. The *C. elegans* connectome intuitively shows a hierarchical dependency structure where information flows from sensory to motor neurons via interneurons.

Treating *C. elegans* connectome with hourglass analysis is different from our previous studies [3, 100], because unlike previous networks which were mostly acyclic, the connectome has many cycles. So, firstly we provide a new framework for applying hourglass analysis on a general digraph. This framework is then applied on the *C. elegans* connectome to uncover that it also possesses a strong hourglass like structure. The corresponding central neurons of the structure are then studied to verify their importance. The connectome of *C. elegans* has two connection mechanism: directed chemical synapses and gap or electrical junctions. We analyze both the chemical synapse network and the combined synapse and gap junction network. On top of the 3-tier hierarchy based on neuron functionality, we explore a finer level of hierarchical structure in the network based on topology to see how neurons within the sensory, inter and motor category are structured. Finally, we show how the hourglass effect is beneficial for architectures like the connectome where many inputs are converted to many outputs through a lot of redundancy.

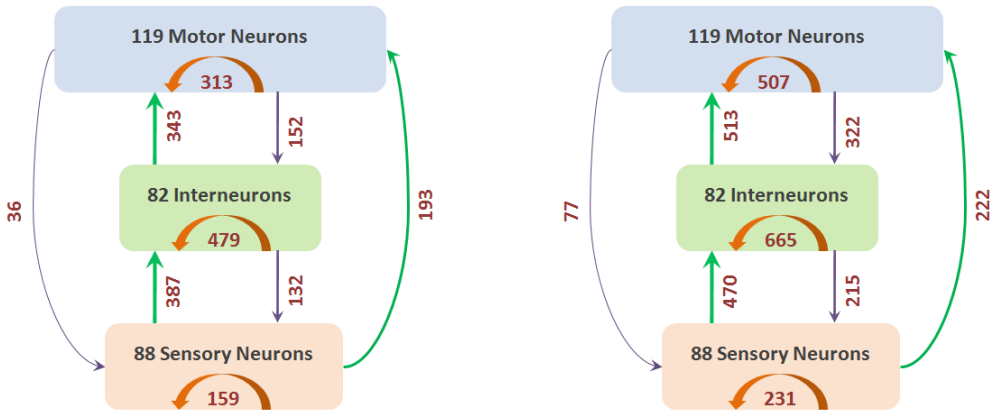
4.2 Basic analysis of the connectome

Our dataset of the hermaphrodite *C. elegans* connectome is derived from [119]. The connectome network is a directed, connected network representing 279 neurons (the 282 non-pharyngeal neurons excluding VC6 and CANL/R, which are missing connectivity data) and 2194 neural connections. These connections are related with 6393 chemical synapses.

We use the terms "connection weight" to describe number of synapses between a pair of neurons and "neuron strength" to represent sum of connection weights entering or leaving a certain neuron. Note that weight does not capture the physical thickness of the synapses. A different type of connection called gap junction or electrical synapse also occurs among neurons. There are 1028 such connections (bidirectional connections between 514 pairs of neurons). Unless mentioned otherwise, we analyze the chemical synaptic connection network throughout the paper (referred to as the synaptic network) and section 4.5 is dedicated for analysis of both synaptic and gap network (referred to as the combined network).

The neurons are classified as sensory (S), inter (I) and motor (M) based on their structure and function [95]. Sensory neurons transfer information from the external environment to the central nervous system (CNS). Motor neurons transfer information from the CNS to external environment and to effector organs (e.g. glands or muscles). Interneurons, process information in the CNS and transfer information from one neuron to the other within the CNS. The network has 88 sensory neurons, 87 interneurons and 119 motor neurons. There are 15 dual-role neurons: 10 behave as sensory-motor, 2 are sensory-inter and 3 are inter-motor. In subsequent analysis, we study information paths originating at sensory neurons and terminating at motor neurons. For purpose of that analysis, we consider the dual sensory-inter and motor-inter neurons as sensory and motor neurons respectively as that provides larger number of candidate information pathways. Similarly for the 10 sensory-motor neurons, we consider them as both sensory and motor. Consequently we end up with 78 sensory neurons, 82 interneurons, 109 motor neurons and 10 sensory-motor (SM) neurons.

The dominant flow of information happens from sensory to motor neurons. As such the connections that flow from S to I, S to M, I to M, SM to M, and S to SM are termed as feedforward (FF) connections. The connections from S to S, I to I, and M to M are termed as lateral (LT) connections. Finally, the connections from I to S, M to I, M to S, SM to



(a) Synaptic network (b) Combined network

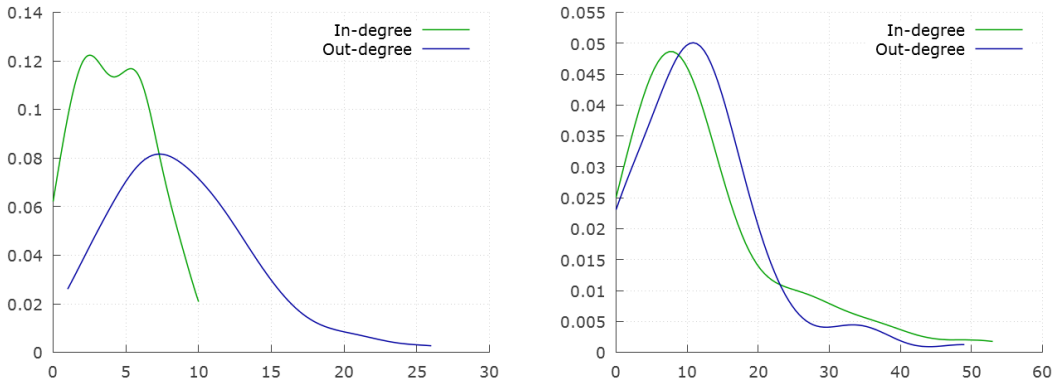
Figure 4.1: Connectome neuron types and number of synaptic and combined connections among them. The synaptic network has 2194 connections and the combined network has 3222 connections.

I, SM to S, I to SM, M to SM, and SM to SM are termed as feedback (FB) connections. There are 923 FF connections, 951 LT connections, and 320 FB connections in the synaptic network.

Figure 4.1 shows the 3 categories of neurons and connections among them for both the synaptic network and combined network. The bulk of the connections are to and from interneurons to other neurons and between interneurons.

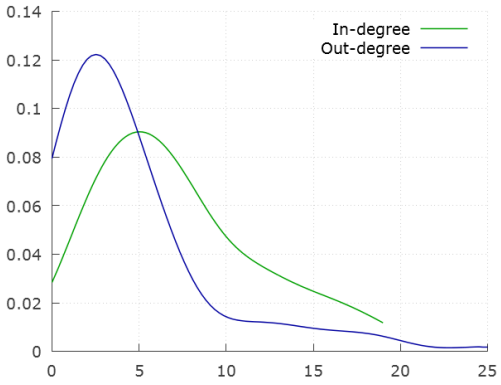
Figure 4.2 shows the in and out degree distribution of the three categories of neurons. Sensory neurons as sources of information flow have smaller in-degrees. Interneurons have the longest tail in their in and out degree distribution. Motor neurons on the other hand have larger in-degrees than out-degrees emphasizing their role as targets of the information flow. Figure 4.3 similarly shows the in and out strength distribution for the three categories of neurons. Both inter and motor neurons show a large in-weight while the sensory neurons have a larger out-weight. This indicates a flow strength gradually from sensory to inter to motor neurons.

Figure 4.4 shows the synaptic weight distributions of FF, FB and LT connections. The



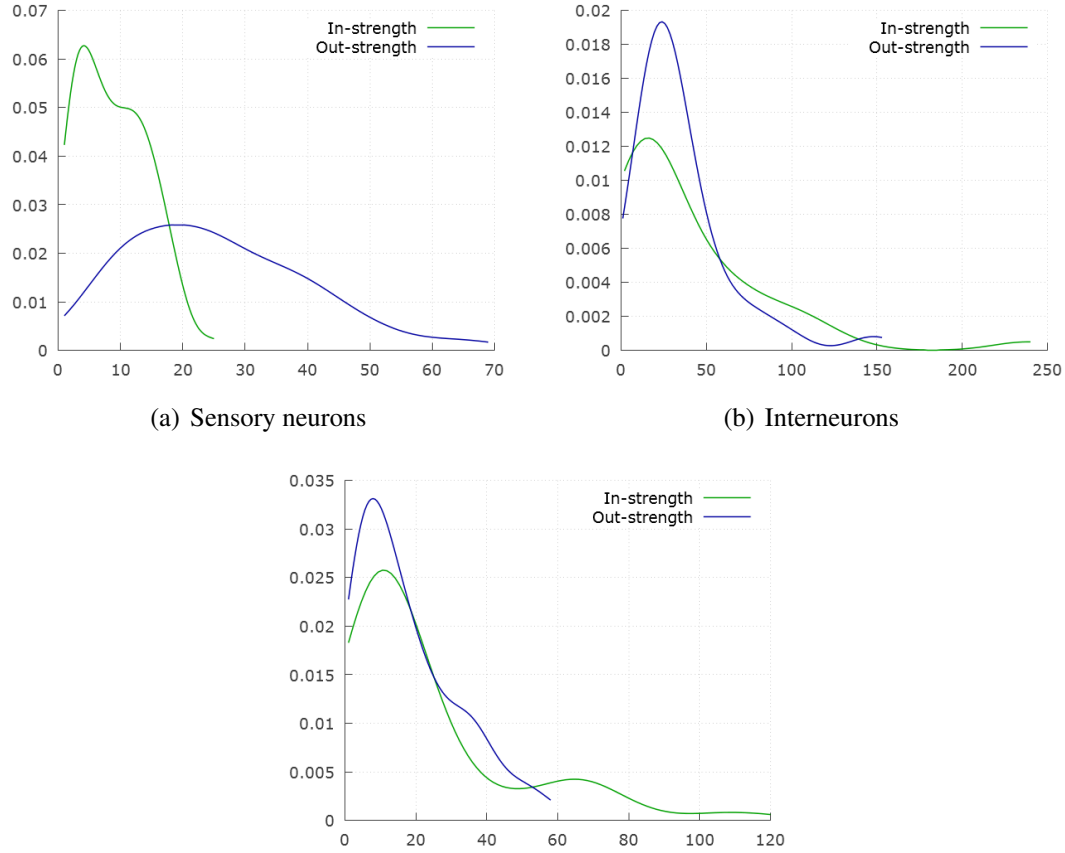
(a) Sensory neurons

(b) Interneurons



(c) Motor neurons

Figure 4.2: In and out degree distribution for sensory, inter and motor neurons.

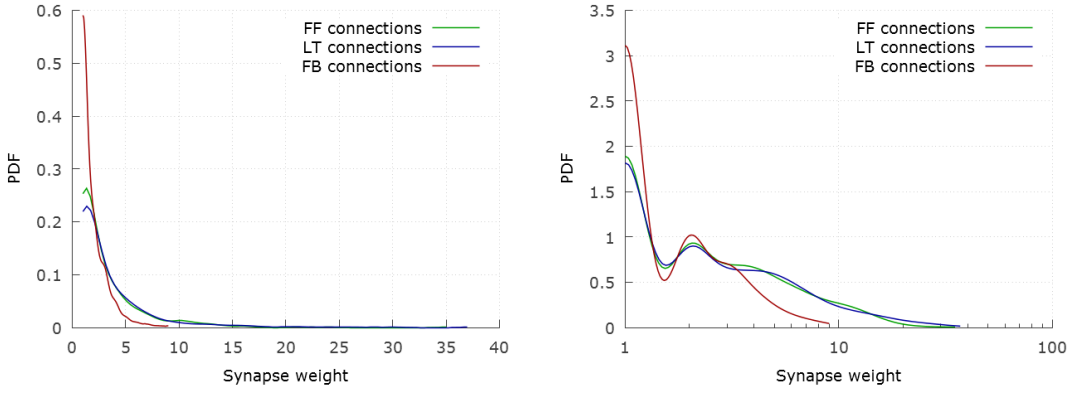


(a) Sensory neurons

(b) Interneurons

(c) Motor neurons

Figure 4.3: In and out strength distribution for sensory, inter and motor neurons.



(a) Linear scale

(b) log-X scale

Figure 4.4: Synaptic weight distribution of FF, LT and FB connections. Weights of FB connections are typically lower than of FF and LT connections.

Table 4.1: Fraction of each category of neuron in the network compared with fraction of that neuron category appearing in the top-20 neurons across degree and strength properties.

Neuron category	Fraction in network	Fraction in top-20 for			
		In-degree	Out-degree	In-strength	Out-strength
S	28%	-	5%	-	5%
I	29%	21%	16%	17%	16%
M	39%	3%	3%	6%	3%
SM	4%	-	-	-	-

distributions mostly overlap in the smaller weight region and FF and LT distribution have a long right tail. This suggests that modulation pathways, which are mostly FB connections have weaker connections.

We took the top 100 synaptic weighted connections to see their distribution of end neurons. All of them were FF connections as expected. For each such type of end neuron distribution, we show the fraction of connections of that type in the whole network and the fraction of connections of that type appearing in the top-100 as follows: S-I:18% & 9%, M-M:14% & 11%, I-I:22% & 3%, I-M:16% & 4%, S-M:8% & 1%. So motor neurons seems to have a stronger tendency of acquiring stronger connections. Of those 100 connections, 52% were FF (5% of all FF connections), 48% were LT (5% of all LT connections) and none were FB.

Table 4.1 depicts which category of neurons have larger degree and strength. Not surprisingly interneurons dominate in both categories signifying their backbone-like role in the network.

4.3 Hourglass analysis of the connectome

The framework of hourglass analysis is based on the notion of dependency paths originating from the source or inputs of a system to the target or outputs of the system. From an information processing perspective, this is analogous to information flowing from sensor

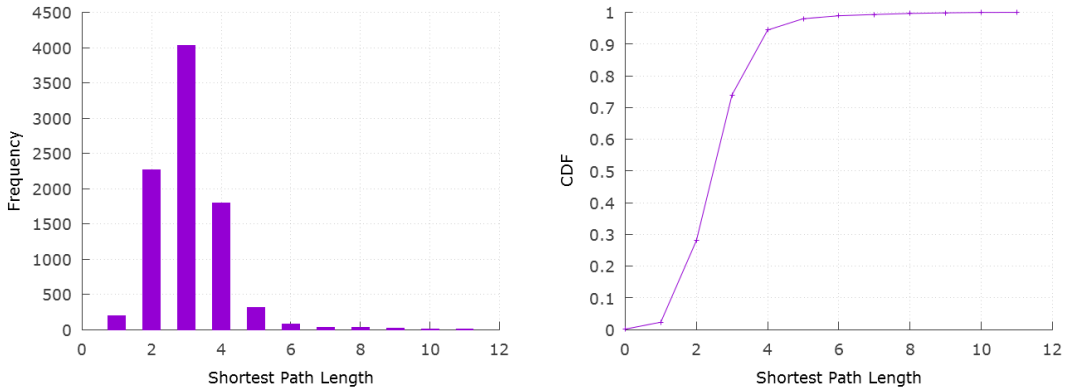
units through intermediary units and finally being utilized at actuator units. In *C. elegans*, outside information is first gathered at the sensory neurons, which is then propagated along other sensory and mostly interneurons along a large number of paths and finally it reaches motor neurons triggering some action. Along with the feedforward flow of information, the connectome network also has strong feedback flow of information demonstrated by the large number of feedback and lateral connections. We will only focus on the feedforward flow in our analysis. We do not know however which are the exact paths that are used for conveying information in this feedforward manner. In our earlier hourglass analysis framework, we assumed all possible feedforward paths from source to target units are equally plausible.

For *C. elegans*, we believe that the underlying routing process of carrying information is efficient and it tends to select shorter paths. Further, it is unlikely that very long paths (in term of number of hops) are utilized, even if they are shortest [17, 53, 92]. Hence, it is also reasonable to assume that FF connections are used for the sole purpose of transferring information from sensory to motor neurons whereas FB connections may be used for regulation via back propagation. Because of the uncertainty in identifying the actual sensor-motor that are utilized in paths used, our approach is to consider a number of plausible the set of paths.

Shortest path based routing. The basic variation will be the set of all shortest paths from each sensory to each motor neuron.

1. "SP": All paths from each sensory to each motor neuron.

Figure 4.5 shows the length distribution of all possible shortest paths from sensory to motor neurons. We see that a maximum length of 5 covers more than 98% of all shortest paths. We use this distribution to create the following path set variations for analysis:



(a) Histogram (b) CDF

Figure 4.5: Length distribution of all possible sensory to motor neurons shortest paths

2. "SP_4": All shortest paths from each sensory to each motor neuron that are at most 4 hops.

3. "SP_5": All shortest paths from each sensory to each motor neuron that are at most 5 hops.

4. "SP^{+1}": All paths from each sensory to each motor neuron that are either shortest or at most one hop longer than the shortest path for that sensory-motor neuron pair.

5. "SP^{+2}": All paths from each sensory to each motor neuron that are either shortest or at most two hops longer than the shortest path for that sensory-motor neuron pair.

6. "SP_4^{+1}": All SP^{+1} paths that are at most 4 hops.

7. "SP_5^{+1}": All SP^{+1} paths that are at most 5 hops.

8. "SP_4^{+2}": All SP^{+2} paths that are at most 4 hops.

9. "SP_5^{+2}": All SP^{+2} paths that are at most 5 hops.

The choice of considering paths that are slightly longer than the corresponding shortest paths is to capture minor deviations from optimality but without sacrificing efficiency.

Optimal path length based routing. Another routing strategy we examine considers all possible paths from sensory to motor neurons up to a certain length. This variation takes us away from the notion of shortest paths and assumes all paths up to a maximum length are likely to be traversed. The considered variations are,

10. "P_4": All paths from each sensory to each motor neuron that are at most 4 hops.

11. "P_5": All paths from each sensory to each motor neuron that are at most 5 hops.

Diffusion based routing. A different approach of routing that does not require global coordination and guidance is a decentralized flow of information in which information disperses through the network across all possible connections [7]. Methods of such information diffusion can be modeled using random walk based models [1], threshold based models [80] etc. We have already considered a constrained form of random walk method for extracting paths that we use where walks are acyclic and up to a maximum hop length (i.e. our P_4, P_5 path sets). A well known threshold-based diffusion model is LTM (Linear Threshold Model) introduced by [42] that works by assigning some vertex of a network a state when some proportion of its neighbors have also adopted that state. Spreading dynamics on regions of the human brain network using LTM was studied by [80]. This diffusion strategy however requires a knowledge of the current state of each region (i.e. network vertices in our case) which we do not have have for the *C. elegans* connectome. Hence we do not employ a general diffusion based routing to extract paths in our study.

Table 4.2 shows relevant properties of the previous 11 path sets. Some key observations from this table are,

- Most pairs of sensory-motor neurons are connected by multiple shortest paths. The number of possible sensory-motor neuron pairs is 9492.

Table 4.2: Properties of the path sets obtained using various routing methods.

	Sensory-motor path sets										
	SP	SP_4	SP_5	SP_4^{+1}	SP_4^{+2}	P_4	SP_5^{+1}	SP_5^{+2}	P_5	SP^{+1}	SP^{+2}
Number of paths	38131	34824	37662	232440	369682	431026	369682	1860756	3162351	401303	3177835
10,50,90 percentile path length	2,3,4	2,3,4	2,3,4	3,4,4	3,4,5	3,4,4	3,4,5	4,5,5	4,5,5	3,4,5	4,5,6
Sensory-Motor pairs connected	92%	87%	90%	87%	90%	87%	90%	90%	90%	92%	92%
Connections used	90%	90%	90%	95%	95%	95%	95%	95%	95%	95%	95%

- The number of possible paths has a 10 fold increase if we allow just one more hop than the shortest paths.
- Most sensory-motor neuron pairs are connected by paths of length of 3-4.
- 8%-10% sensory-motor neurons pairs are not connected in any of the path selection strategies. This might be caused by two reasons:
    1. Due to removal of feedback connections, the connectivity between some pairs was lost. This do not however affect our analysis based on feedforward information flow.
    2. Some pairs of sensory-motor neuron may not have communication pathways anyways based on the connectome topology.
- 5%-10% connections are not used in any path sets. It is likely that those connections are only used in feedback flow of information since some of the FF/LT connections may also be used in feedback paths.

Figure 4.6 shows the distribution of path centrality (the number of paths passing through a vertex) for all neurons across different path sets. All distributions indicate the existence of a small number of neurons through which most communication paths go through.

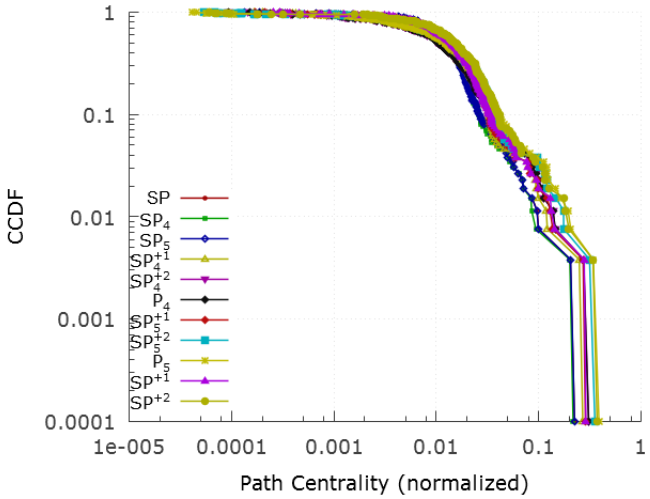


Figure 4.6: Distribution of path centrality among neurons across all path selection strategies.

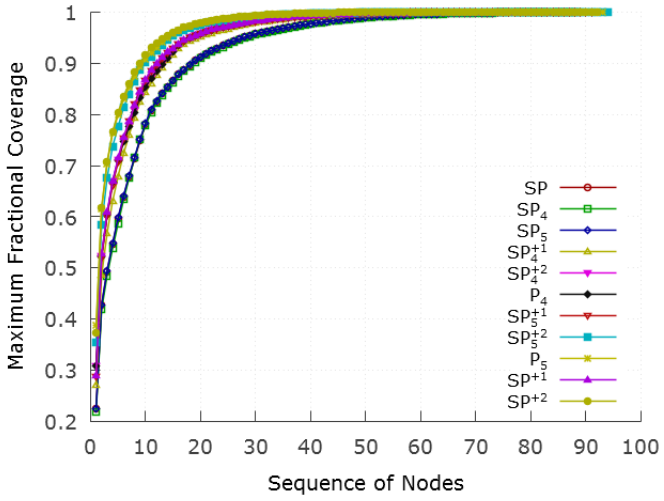
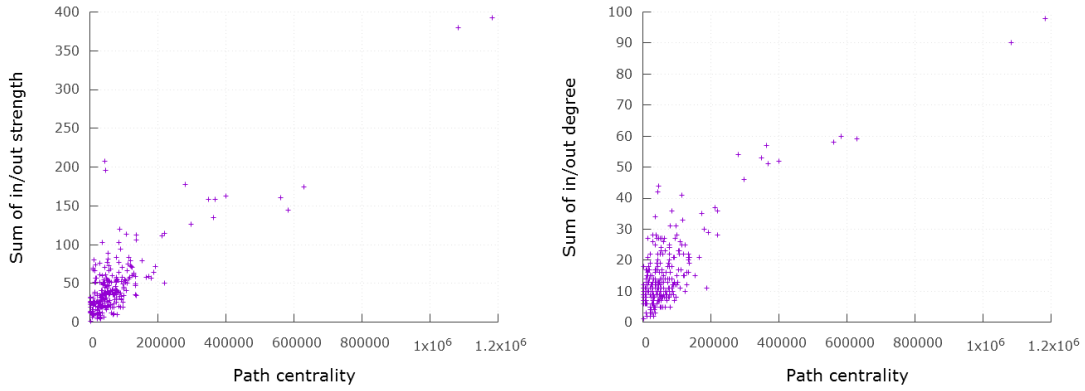


Figure 4.7: The maximum path coverage for various path sets.



(a) Centrality and strength. Pearson correlation coefficient 0.84, P¡0.0001, 95% CI 0.80 to 0.87

(b) Centrality and degree. Pearson correlation coefficient 0.80, P¡0.0001, 95% CI 0.77 to 0.85

Figure 4.8: Scatter plot of path centrality and (a) total strength and (b) total degree. Path set used in SP^{+2}.

Figure 4.7 shows the cumulative path coverage (path coverage of set of vertices is the fraction of total paths that pass through some vertex of that set) as a function of the number of vertices in the core, for each path set. All path sets demonstrate a sharp knee at around 90% of path coverage with a small number of vertices. A slight deviation (90% coverage requiring more vertices than others) is observed for the three strictly shortest path based strategies. These routing strategies have much smaller number of total pathways compared to the other routing strategies and hence their core vertices do not accumulate as much coverage.

Figure 4.8 shows the relationship between path centrality with their total strength (i.e. sum of in and out strengths) and total degree (sum of in and out degree). We have used SP^{+2} as the representative path set. Both have a positive correlation with centrality; strength seems to be stronger in correlation. Although neurons having large degree are expected to participate in more paths, neurons with larger number of synapses connected to them also appear to take part in more communication paths. This is an interesting result since paths were computed ignoring synaptic weights.

Figure 4.9 examines the effect of the path coverage threshold τ (the path coverage thresh-

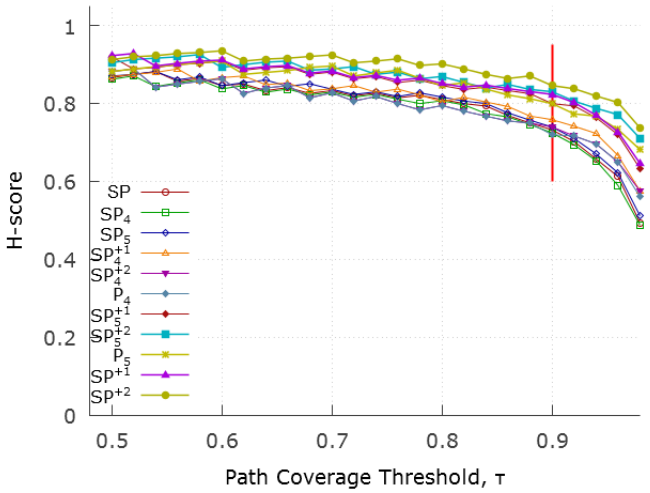


Figure 4.9: Effect of path coverage threshold, τ on H-score for various path sets.

old determines what fraction of total paths should be covered by vertex set to be considered as the network core) on the resulting H-score (H-score or hourglass score quantifies the extent of a network's similarity to hourglass structure) for each path set. In all cases, the H-score is close to one (i.e. highly hourglass like) and stable for threshold values up to 90% (marked with a red line), after which the core size increases significantly to cover the last remaining paths and hence a sharp drop in H-score is observed. This observation is similar to our prior work [100] on other types of networks.

Table 4.3 shows the sequence of core neurons (for $\tau = 0.9$) for each path set. We can see 8-10 highly conserved neurons that top as the most important core neurons across all path sets. This suggests that the key outcome of our analysis framework is robust against the choice of path sets. We see that the size of the core decreases as we move towards path sets containing larger number of paths. This indicates the hourglass effect becomes stronger with more paths (see corresponding H-scores) which is caused by existence of a strong and small core vertex set. We can also see a symmetry in the information flow as the core contains both the left and right hemisphere of most neurons. We also have a column showing the "Rich-club" vertices extracted from [115]. A rich-club is a subgraph of high-degree vertices that are densely interconnected among each other above and beyond what

Table 4.3: Core neurons appearing from hourglass analysis for each path set. The core is computed for $\tau = 0.9$. Numbers corresponding to neurons for a path set represent what fraction of the 90% paths was covered by that neuron.

Core Neuron	SP	SP_4	SP_5	SP_4^{+1}	SP_4^{+2}	P_4	SP_5^{+1}	SP_5^{+2}	P_5	SP^{+1}	SP^{+2}	Rich-club
AVAL	0.25	0.24	0.25	0.24	0.24	0.24	0.26	0.25	0.25	0.26	0.27	✓
AVAR	0.23	0.22	0.23	0.30	0.34	0.34	0.32	0.40	0.43	0.32	0.41	✓
AVBL	0.08	0.07	0.08	0.10	0.09	0.09	0.09	0.10	0.10	0.09	0.10	✓
AVBR	0.04	0.04	0.04	0.04	0.03	0.03	0.04	0.03	0.03	0.04	0.03	✓
PVCL	0.05	0.05	0.05	0.07	0.07	0.07	0.07	0.07	0.07	0.06	0.06	✓
PVCR	0.03	0.03	0.03	0.02	0.02	0.02	0.02			0.03	0.04	✓
AVEL	0.06	0.06	0.06	0.05	0.04	0.04	0.05	0.04	0.03	0.05	0.03	✓
AVER	0.06	0.06	0.06	0.06	0.05	0.05	0.06	0.05	0.04	0.05	0.04	✓
AVDL	0.02	0.02	0.02	0.02	0.01		0.01					✓
AVDR	0.04	0.04	0.04	0.04	0.03	0.03	0.03	0.02	0.02	0.03	0.01	✓
DVA	0.05	0.05	0.04	0.02	0.03	0.03	0.03	0.03	0.02	0.04	0.01	✓
HSNL	0.01		0.01									
HSNR	0.02	0.03	0.03	0.02	0.02	0.01	0.01	0.01		0.02		
RIAL	0.01	0.02	0.01	0.01	0.02	0.02	0.01		0.01	0.01		
RIAR	0.01	0.02	0.01			0.01						
RIMR	0.01	0.01	0.01	0.01	0.01	0.02						
RMGL	0.01	0.01	0.01									
PVR	0.01	0.01	0.01									
AIBR	0.01	0.01	0.01									
AIZL		0.01										
H-score	0.74	0.73	0.74	0.76	0.74	0.72	0.80	0.83	0.80	0.82	0.85	

would be expected on the basis of their degrees [26] and have been shown to play important role in the human brain connectome [117] as well as *C. elegans* connectome [115]. We will discuss more about the rich-club in the next section.

For all subsequent analysis, we will use the "SP^{+2}" path set. The choice of SP^{+2} is based on the following reasons:

- Paths longer than 4-5 are rarely found in the well-known neural circuits (we discuss a set of such neural circuits in the next section). But rather than using a global cutoff length, we use the shortest path length for each sensory-motor neurons pair as a cutoff estimate for finer granularity.
- Exclusive shortest path based routing schemes have faced criticism over its plausibility. Shortest path algorithms require global topological knowledge across all vertices or local topological knowledge sharing among neighbor vertices, both of which is highly unlikely in a physiological system [6]. Moreover, it has been shown that reliance on strict shortest path based routing renders a large number of connections unutilized, another unlikely phenomenon for a highly efficient and evolved structure like the brain network [7].
- SP^{+2} adequately captures best of both worlds by favoring shorter paths but not demanding computation of the shortest paths. Rather it can be well explained by a diffusion-based approach that is limited by the distance traveled due to loss of signal intensity/integrity.

4.4 Importance of core neurons

We now explore the role of the identified core neurons based on the SP^{+2} set of paths. There are 10 neurons in the core: 7 of which are located in the head region (AVAR/L,

AVBR/L, AVER/L, AVDR) and 3 are located in the tail region (PVCR/L, DVA). If we want to extend the set of core neurons slightly by requiring to cover 95% of all paths instead of 90%, we add 4 more neurons to the set, (HSNR, AVDL, RIAL, RIMR). The original 10 core neurons contain 9 command interneurons that play a pivotal role in the forward and backward locomotion [115]. The one missing command interneurons (AVDL) appears in the extended core neurons set (for τ=95%). The other non-command interneuron of the core, DVA, is known as a proprioceptive interneuron modulating the locomotion circuit [115].

In Table 4.4, we have collected a number of functional circuits from the *C. elegans* literature to identify the functional role of our core neurons. The core neurons are involved in many functional circuits requiring spontaneous and planned movement. The central task associated with most core neurons is locomotion, and for an organism of this minuscule size locomotion usually precedes or enables most other tasks. As such, the importance of the core neurons is central to the organism. Topologically, the core vertices of an hourglass network integrate and compress input information, and encode functions that are reused across multiple outputs [107]. Many of the adaptive behaviors of the organism such as feeding, egg-laying, escape and navigation result from this functional integration performed by the core neurons. Some of the circuits shown (e.g. thermotaxis, chemosensation, olfactory behavior etc.) perform tasks mostly involving sensory neurons and usually followed by a locomotory response involving the core neurons. The interneuron RMG seen as important for local environment assessment for example, mostly interacts with sensory neurons by feedforward and feedback signals and did not make it the core due to lack of direct/indirect interaction with motor neurons. A few other interneurons like, RIA and RIM that are seen in many circuits belong to our extended core set. An important set of interneurons (known as amphid interneurons) AIA, AIB, AIY, AIZ also has large interactions with mostly sensory neurons in most circuits and in turn passes on the collected signal to the core neurons

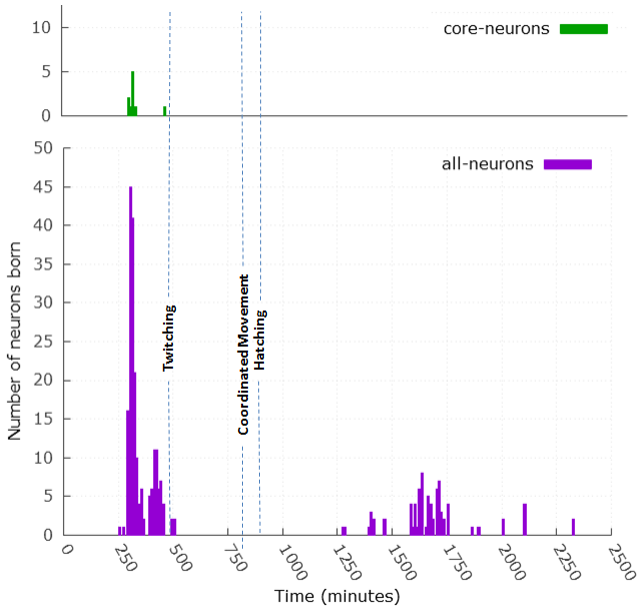


Figure 4.10: Contrast of birth times of hourglass core neurons versus all neurons of *C. elegans*. The dashed vertical lines shows key development events. Figure adapted from [115].

making them take locomotory responses. A couple of these amphid neurons appear in the strictly shortest path based cores which were bigger in size as shown in Table 4.3.

From Table 4.3, we can also see a substantial overlap of our core neurons with the rich-club neurons identified by [115]. This is an interesting outcome, since the topological analysis employed in that study is completely different (it depends on vertex degrees, while ours depends on sensory-to-motor paths). This emphasizes the importance of these core neurons. Some of the other key features of their analysis naturally applies to core neurons, e.g. birth times of the core neurons. Figure 4.10 shown the time when core neurons start developing after fertilization contrasting with birth times of all other neurons. All core neurons were developed early and all of their neuronal components were formed before the first visible signs of motor activity (twitching) [115].

Table 4.4: Notable functional circuits and their participating neurons in *C. elegans* connectome. Core neurons from hourglass analysis are shown in red-italic.

Functional Circuit	Sensory neurons	Interneurons	Motor neurons
Tap-withdrawal [70]	PVD, PLM, AVM, ALM	*AVA*, *AVB*, *AVD*, *PVC*, *DVA*	FWD, REV
Touch sensitivity [23]	ALM, AVM, PLM, LUA	*AVA*, *AVB*, *AVD*, *PVC*	AS
Thermotaxis [62] – isothermal, negative [38]	AFD, AWS, ASI	AIB, AIY, AIZ, RIA, RIB, RIM	
Chemosensation [11]	Many		
Mechanosensation [39] – avoidance, foraging, – light-harsh touch, tap [97]	AVM, ALM, PVD, PLM, ASH, PDE, ADE, CEP	*AVA*, *AVB*, *AVD*, *PVC*	VA, VB, DA, DB
Egg laying [123]			HSN, VC1-6
Defecation [8]		AVL, DVB	
Feeding [8]		AVL, DVB	
Olfactory behavior [22] – odor removal	AWS	AIA, AIB, AIZ, AIY	
Locomotion [32]		*AVA*, *AVB*, *AVD*, *PVC*	VA, VB, VD, DA, DB, DD
Repulsive motion [44]	ASH	*AVA*, *AVB*, *AVD*, AVE, *PVC*, RIM	
Local environment assessment [73, 76]	ADL, ASH, AWB, ASK, IL2, URX	RMG	
Navigation [43]	AWC, ASI, ASK	AIA, AIB, AIZ, AIY, RIA RIM, RIB, SMD, SIA, SIB	RIV, RMD, SMV
Muscle contraction, locomotion modulation [72]		*DVA*	
Locomotory state regulation [122]		AIA, AIB, AIZ, AIY	
Escape response [89]	ALM, AVM, PLM, PVM	*AVA, AVD, AVB, PVC*, RIM, VB, DA, DB	SMD, RMD, VA,
Proprioception [49, 103]	PVD	*DVA*	

4.5 Gap junction analysis

Gap junctions provide a different type of connectivity between neurons than chemical synapses. Chemical synapses use neurotransmitters to provide neuronal connectivity, while gap junctions work by creating electrical channels to provide faster neuronal coupling. The directionality of the current flow in gap junctions can not be detected from electron micro-graphs. Hence they are treated as undirectional (or bidirectional) in the *C. elegans* connectome.

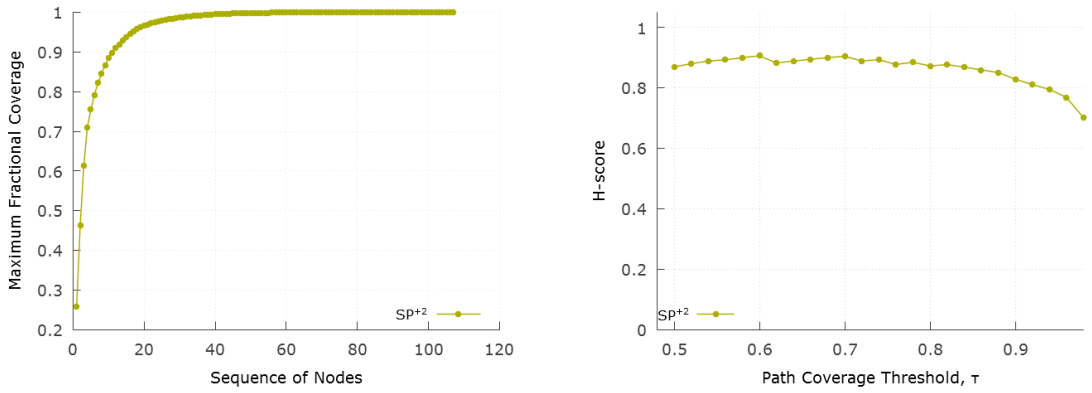
We consider the gap junction network in conjunction with the chemical synapse network and refer to it as the combined network. Some properties from the analysis of the combined network:

1. The gap junction network consists of 253 neurons and 1028 directed connections. We count each undirected connection as two directed connections. There are 514 such undirected connections.

2. 64% of the gap junction network connections do not co-occur with any chemical connection (i.e. they are not present in the synaptic network), while 83% of the synaptic network connections do not co-occur with any gap junction connection.

3. The combined network consists of 3222 (2194+1028) directed connections and we remove feedback connections in the same way we did for the synaptic network. That leaves us with a network of 279 neurons and 2608 connections. That network has 1205 FF, 1403 LT, and 614 FB connections.

4. There are 7698071 SP^{+2} paths (paths having at most 2 extra hops than the corresponding shortest paths for each pair of SM neurons) in the combined network which is approximately 2.5 times more than the corresponding synaptic network paths.



(a) Maximum cumulative path coverage (b) Effect of path coverage threshold, τ on H-score

Figure 4.11: Hourglass analysis for combined network with SP^{+2} set of paths used.

Table 4.5: 12 core neurons from the hourglass analysis of the combined network. 10 neurons (shown in red-italic) were also the core of the synaptic network.

Neuron	*AVAL*	*AVAR*	*AVBL*	*AVBR*	*AVEL*	*AVER*	*PVCL*	AIBR	*AVDR*	*DVA*	*PVCR*	VD01
Coverage Contrib.	0.29	0.23	0.16	0.11	0.05	0.04	0.03	0.03	0.02	0.02	0.01	0.01

Figure 4.11(a) shows the cumulative path coverage as a function of the number of vertices in the core. Figure 4.11(b) examines the effect of the path coverage threshold τ on the resulting H-score. Both results are similar to what we have seen for the synaptic network and demonstrate strong hourglass-like architecture.

With τ=0.9, the resulting core vertices are shown in Table 4.5. The H-score for the combined network is 0.83.

The two new neurons that appear in the combined core are:

- AIBR: inter - locomotion, food and odor evoked behaviors, local search, lifespan and starvation response
- VD01: motor - Sinusoidal body movement-locomotion.

4.6 Hierarchical structure of the connectome

We have been analyzing the *C. elegans* connectome so far by focusing on the feedforward flow of information from sensory to motor neurons via interneurons. This flow establishes a natural 3-tier hierarchy. In this section, we further investigate the hierarchical structure of the network. The goal of this analysis is to obtain a DAG representation in which each vertex is placed at an unique hierarchical level, based on the vertex ordering information provided by the source-target paths. We want to examine the depth of this hierarchy, how many of each of the 3 classes are placed at each hierarchical level, whether the core neurons are located at the same or different level of the hierarchy etc.

4.6.1 Hierarchy inference framework

We first propose the following general framework to infer hierarchy from a directed network based on a given set of source-target paths. We are given a directed network G. Suppose that the vertices of G are placed at a layered hierarchy H (the "ground-truth hierarchy"). Vertex v is placed at layer $h(v)$. An connection to a higher layer vertex is referred to as *feedforward*, an connection to a same layer vertex is referred to as *lateral*, and an connection to a lower layer vertex is referred to as *feedback*. The feedforward flow of information is assumed to be from lower layers to higher layers, while the feedback flow of information is from higher layers to lower layers. Both flows of information can traverse lateral connections.

We are also given a set of *source-target feedforward paths* P on G. These paths originate at vertices that we refer to as *sources* and terminate at vertices that are called *targets*. The constraint is that all paths in P traverse only feedforward and lateral connections – they do **not** traverse feedback connections. Note that there can be connections from one source

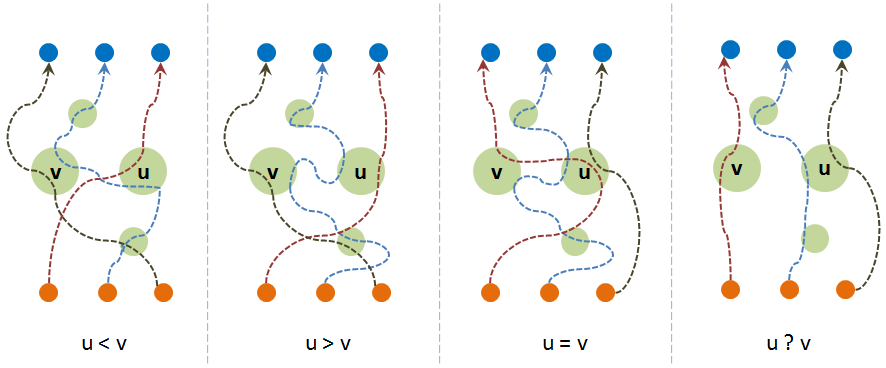


Figure 4.12: Hierarchy relationship between two vertices based on the paths traversing them.

vertex to another source vertex (and similarly for targets).

The main premise of this work is that we can infer the hierarchy H (i.e., to infer the layer of each vertex) as long as we are given a sufficiently large set of paths P. The reason is that paths traverse vertices in an order that is consistent with the underlying hierarchy H. So, if there are paths that traverse vertex u before they traverse vertex v, but there are no paths that first traverse v and then vertex u, it is reasonable to infer that u is at a lower layer than v. On the other hand, if there are also paths that first traverse v and then u, it is reasonable to infer that u and v are at the same layer.

Let $p_{u,v}$ be the number of paths in P that first traverse u and then v. Two vertices u and v can have one of the following relations (illustrated in Figure 4.12):

1. $p_{u,v} > 0$ and $p_{v,u} = 0$: We infer that u is at a lower layer than v and write $u < v$.

2. $p_{u,v} = 0$ and $p_{v,u} > 0$: We infer that u is at a higher layer than v and write $u > v$.

3. $p_{u,v} > 0$ and $p_{v,u} > 0$: We infer that u is at the same layer than v and write $u = v$. Note that the relative magnitude of $p_{u,v}$ and $p_{v,u}$ does not matter because the number of paths in the two directions may be quite different.

4. $p_{u,v} = 0$ and $p_{v,u} = 0$: We cannot infer anything about the relation between u and v

and write $u?v$.

We can now pose the following optimization problem to construct a hierarchy H' that is as close as possible to H: *Given the set of paths P, construct a hierarchy H' so that the number of distinct feedforward connections traversed is maximized, subject to the constraint that no path in P traverses a feedback connection. Traversing lateral connections is allowed but it does not contribute to the objective function.*

Let us first construct a directed network, G' of pairwise relations between vertices. When $p_{u,v} > 0$, it may be that $u < v$ or $u = v$, and so we write $u \leq v$. We can represent this with a directed connection from u to v in the *network of relations*. So this connection would be missing from that network either when $u?v$ or when $u > v$. Then, we can examine if there are any Strongly Connected Components (SCCs) in this network of relations. If there is an SCC that involves vertices $(v_1, \ldots, v_k)$ it means that each of these k vertices is "lower or equal" than any other vertex in that set. The only way to avoid a relational conflict is by setting all these relations to "equal" – and thus placing all these vertices at the same layer.

Algorithm 1 describes our proposed heuristic method for inference estimation. We begin by constructing relationship network between each pair of vertices. After that we construct the hierarchy bottoms up one layer at a time. In each layer we include vertices that are hierarchically lower, equal or unrelated to all other remaining vertices. In case of equality, all vertices belonging to a SCC are placed in the same layer. After constructing each layer, we remove all vertices placed at that layer from the network of relations. Note that the network of relations and the SCC of each vertex can be computed incrementally – they do not need to be recomputed from scratch every time. We illustrate the working of the algorithm in Figure 4.13.

Algorithm 1 Hierarchy Inference Algorithm

$G'(V', E') \leftarrow$ the network of relations with vertices V' and connections E' ▷ Input
$currentLayer \leftarrow 1$
while V' is not empty **do**
    for each vertex $v' \in V'$ **do**
        let $S(v')$ is the SCC containing v' in G'
        if ($v' < w$ or $v'?w$) for all $w \notin S(v')$ **then**
            assign all vertices of $S(v')$ to $currentLayer$
        end if
    end for
    remove all vertices placed at $currentLayer$ from G'
    $currentLayer = currentLayer + 1$
end while

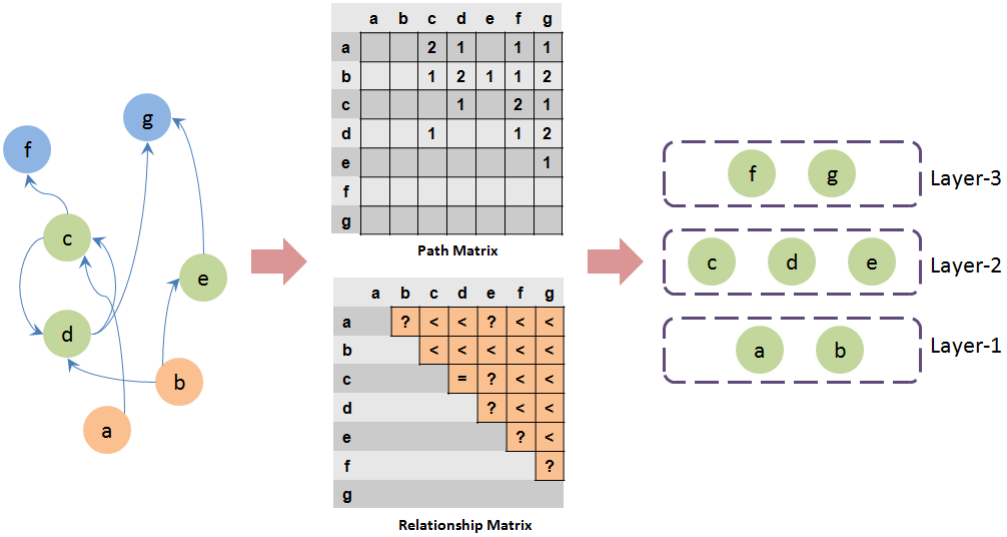


Figure 4.13: Illustration of the hierarchy inference algorithm: we begin with the network of the left. Considering all paths from sources (a, b) to targets (f, g), we create the path matrix (i.e. the matrix of $p_{u,v}$ for $\forall u, v$ pairs). Then using the path matrix we generate the relationship network, represented with the bottom matrix in the middle. We show the "= and ¡ and ?" relationships in the matrix that result in a bidirectional, unidirectional, and no connection in the relationship network respectively. Consider vertices c, d, e in the relationship network. c, d form a SCC (hence the = relationship), are positioned below vertices f, g, above vertices a, b and unrelated to vertex e. Hence all the three vertices c, d, e are placed at layer-2.

Table 4.6: Properties of the connectome's inferred hierarchy for SP^{+2} path set.

Hierarchy level	#Neurons	#Sensory neurons	#Interneurons	#Motor neurons	#Core neurons	#SCCs	Max SCC size
1	21	21	0	0	0	21	1
2	32	32	0	0	0	6	25
3	24	22	2	8	0	21	3
4	6	5	0	2	0	4	3
5	4	3	1	0	0	2	3
6	3	3	0	0	0	2	2
7	64	1	63	0	10	3	62
8	56	1	1	55	0	20	35
9	11	0	0	11	0	7	5
10	15	0	0	15	0	4	12
11	3	0	0	3	0	3	1
12	2	0	0	2	0	2	1
13	1	0	0	1	0	1	1
14	3	0	0	3	0	3	1
15	10	0	0	10	0	2	9
16	2	0	0	2	0	2	1
17	6	0	0	6	0	4	2

4.6.2 Hierarchy of the connectome

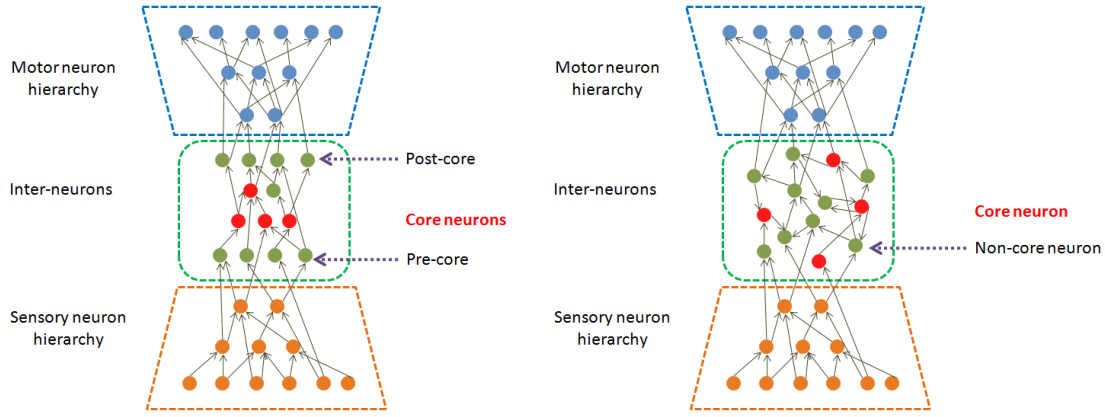
Table 4.6 shows the hierarchy obtained by running the heuristic algorithm on the *C. elegans* network. We see the following key properties of the hierarchy:

- The network hierarchy has a depth of 17 layers. Sensory neurons are mostly below the interneurons, and the motor neurons are mostly above the interneurons. The sensory and motor neurons have similar hierarchical structure. The interneurons, however are predominantly at the same layer. The width of the layers gradually shrinks within each of the 3 categories of neurons as expected.
- The core neurons are not hierarchically structured among themselves, all of them appear in level 7 alongside most other interneurons indicating they are all mutually connected.

We have verified the robustness of the inferred hierarchy against all the path selection strategies mentioned earlier. The key features of the hierarchy: separation between sensory,

Table 4.7: Key properties of the inferred connectome hierarchy across all path sets.

Path set	#Layers in hierarchy	Layer number(s) containing core neurons	Layer number(s) containing most ($> 95\%$) interneurons
SP	12	3	3
SP_4	10	2	2
SP_5	10	2	2
SP_4^{+1}	12	3	3
SP_4^{+2}	13	3	3
P_4	13	3	3
SP_5^{+1}	13	3	3
SP_5^{+2}	17	7	7
P_5	17	7	7
SP^{+1}	13	3	3
SP^{+2}	17	7	7



(a) Hierarchical core (b) Non-hierarchical core

Figure 4.14: Example of a hierarchical and a non-hierarchical core in two hourglass networks.

inter and motor neurons hierarchy and most of the interneurons (including core neurons) belonging to the same layer, is found to be conserved across all the path sets. We show these properties in Table 4.7.

Even though we have a small core (only 10 neurons), these interneurons are at the same hierarchical level with many other interneurons. So, even though most paths are covered by a small number of core interneurons, these interneurons communicate through lateral connections with many more interneurons. We illustrate this case in Figure 4.14 where we show an hourglass structure with or without a hierarchical core. The figure on the left

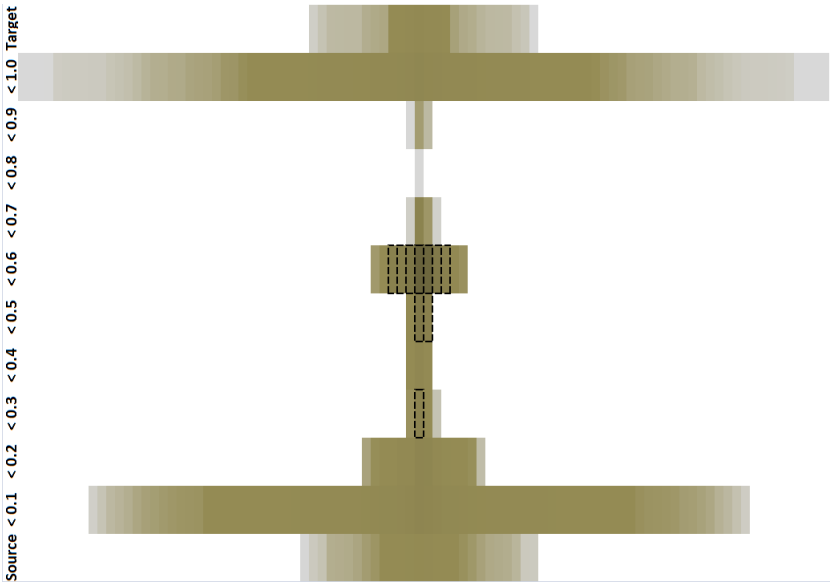


Figure 4.15: Visualization of the *C. elegans* connectome based on the "Location" metric. For details of this visualization, please refer to Figure 2.7.

shows an hourglass where the core neurons are grouped in a compact hierarchy and there are interneurons clearly above and below the core neurons. This however is not the case for the *C. elegans* network. We see all the interneurons at the same hierarchical level. This indicates a situation where the interneurons are strongly connected and not hierarchical. This however does not influence the fact that some of the interneurons were strategically connected and traversed by a large number of sensory-motor paths. This situation is illustrated in the right example network.

Earlier we developed the "location" metric for visualizing the vertex hierarchy of a network in a vertical orientation. We compute the location metric on the connectome to find the relative position of the neurons and show it in Figure 4.15. Note, how the core neurons (marked with dotted border) position themselves in the middle of the hierarchy and vertices in the middle of the hierarchy are the highest path centrality ones.

4.7 Dimensionality Reduction

In this section we seek to understand what benefit a network gains by having an hourglass architecture. We focus on systems that produce a set of outputs from a set of inputs where the inputs have some redundancy (their information can be compressed), and the outputs are not completely independent either meaning that they can be produced by reusing some intermediate components. For such systems, we show how having an hourglass architecture can be beneficial with respect to aggregated cost of constructing all the outputs. We first define a framework for quantifying the construction costs of the outputs of a system. Then we apply the framework on the *C. elegans* connectome and analyze the cost benefit it provides by having an hourglass structure.

4.7.1 Cost analysis framework

The systems we analyze are dependency networks that have designated sets of source, intermediate and target vertices. Sources are the elementary building blocks. Every path a target vertex has from some source vertex signifies an unique way that source is contributing to the construction of that target. We explain this concept using a Boolean circuit system shown in Figure 4.16. Consider the output signal t_1. It can be produced using the input vertices only (not using the intermediate vertices) with the expression $(\overline{a} \oplus (a + b + (c \oplus d)))$. Notice that the expression contains 5 operands where each indicates one way the operand which is an input vertex (or signal) is involved in producing t_1. Each of these 5 operands can be traced back to its corresponding input vertex from t_1 using a path in the circuit system. In other words, we see that the number of paths (or dependencies) a target vertex has from source vertices indicates the cost of producing that output without using any intermediate vertices. Now consider the target vertex t_1 again, but this time constructed using the intermediate vertices given by the expression $(i_1 \oplus i_2)$. t_1 depends on intermediate vertices

i_1, i_2 only and they compressed the 5 input signals coming from the input vertices a, b, c, d by turning them into 2 output signals. We see that cost of constructing t_1 goes down to 2 operands, from the earlier cost 5 operands when build entirely from inputs. We call this the reduction of dependency dimensionality of target t_1 using intermediate vertices $i1, i2$. For a target, if dependencies from a large number of inputs is reduced/compressed to a small number of intermediate vertices as in an hourglass architecture, we gain a large reduction of dependency dimensionality. However, the small number of intermediate vertices may not entirely compress all the dependencies a target has. Consider target t_2 for example. It has a dependency from input b, which can not be compressed by any intermediate vertex. Therefore we also need to consider if the dimensionality reduction covers all the input dependencies of a target to measure its effectiveness.

We now formally define the metrics for computing the cost benefit in terms of dimensionality reduction of a dependency network. For a target t, the cost of its construction is proportional to the number of source vertices it depends on. Let us denote the set of source vertices t depends on as S_t. Now let us assume this cost can be "compressed" to a small number of intermediate vertices, which we call "core vertices, C". The subset of core vertices that t depends on is denoted by C_t. We define the dimensionality reduction for t as $(S_t - C_t)$. This signifies that the dependencies t had from the elementary sources can reduced to smaller a cardinality of intermediate vertices but keeping t still fully functional. To ensure the last fully functional condition, it must be the case that t can be constructed using C_t only. If not, the strength or "goodness" of the dimensionality reduction is reduced. We compute the goodness of dimensionality reduction for t as follows: the number of path t has from sources indicated the possible ways source vertices influence the final construction of t. Let us call this number of paths P_{S_t}. However of these paths, the ones that go through C_t have their influence compressed into C_t. Let call this later number of paths P_{C_t}. Then the goodness of dimensionality reduction t can achieve with C_t is $\frac{P_{C_t}}{P_{S_t}}$. This

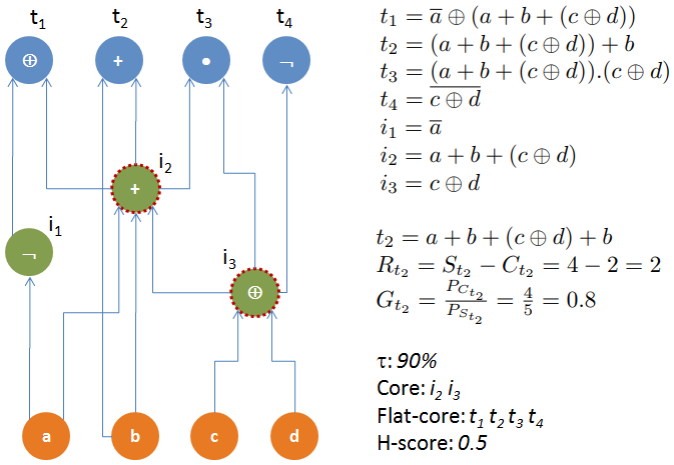


Figure 4.16: The network has 4 input signals (i.e. source vertices): a, b, c, d, 3 intermediate circuit components: i_1, i_2, i_3 and 4 output signals (i.e. target vertices): t_1, t_2, t_3, t_4, along with 14 edges. The number of paths that connect each target vertex to a source is 22, $(t_1 : 5, t_2 : 5, t_3 : 6, t_4 : 6)$. We assume vertices i_2, i_3 as cores. Target t_2 for example depends on sources a, b, c, d and cores i_2, i_3. So its dimensionality reduction is 2. The number of paths t_2 has from sources is 5, out of which 4 go through C_{t_2}. So the goodness of dimensionality reduction for t_1 is $\frac{4}{5} = 0.8$. Note that the final expression for t_2 requires 5 operands and 4 operations. The number of final operation needed for computing a target is proportional to the number paths that terminate at it (e.g. 5 paths for t_2) or equivalently number of operands it has in the souce-only expression. For the remaining target vertices, dimensionality reduction and goodness of reduction are $t_1 : 1; 0, t_3 : 2; 1.0, t_4 : 1; 1.0$.

signifies the effectiveness of the compression by the core vertices in capturing the source dependencies.

Dimensionality reduction for a Boolean circuit system is illustrated in Figure 4.16. In this example we assume source to target flow traverse all possible paths.

4.7.2 Dimensionality reduction analysis of the connectome

We now apply the dimensionality reduction analysis on the *C. elegans* network. Figure 4.17 shows for each motor neuron of the *C. elegans*, the amount of reduction it achieves with the 10 core neurons. Almost all the neurons has a large difference between the number of

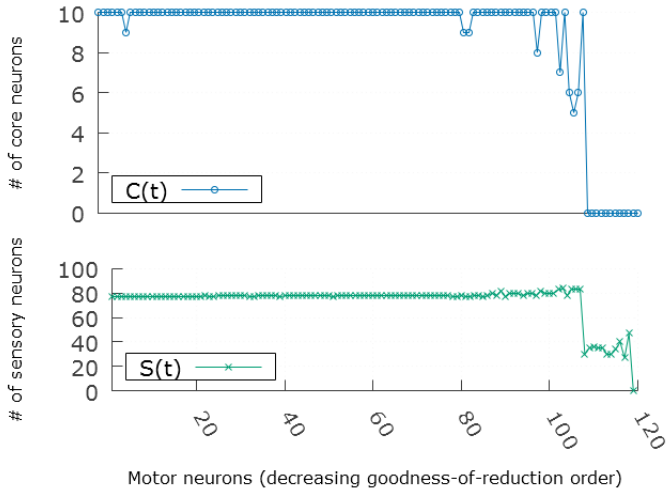


Figure 4.17: Dimensionality reduction of target (motor) neurons in *C. elegans* network shown in decreasing goodness of cost reduction order.

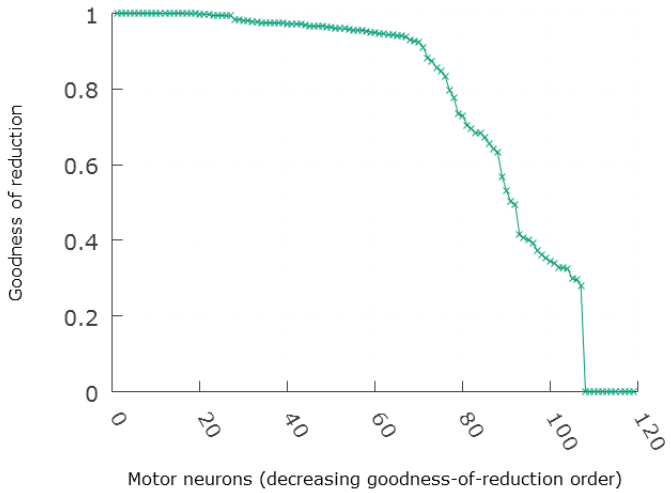


Figure 4.18: Goodness of cost reduction of target (motor) neurons in *C. elegans* network shown in decreasing order.

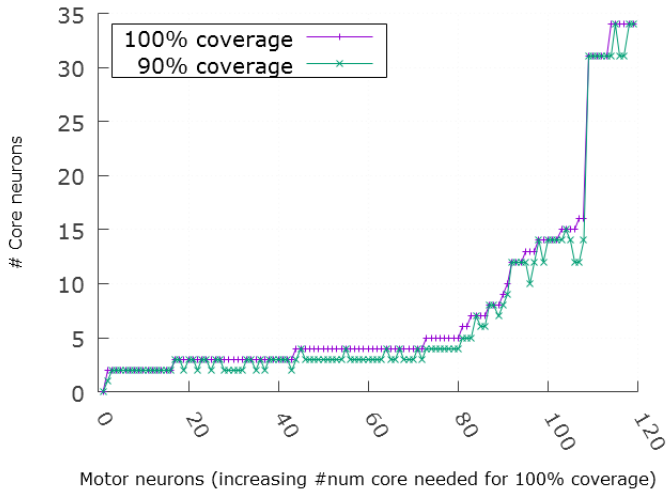


Figure 4.19: Target (motor) neurons in *C. elegans* network are shown in increasing order of the number of core vertices needed to cover all paths terminating at them.

sensory neurons it depend on and the number of core neuron it depends on.

However to verify whether all of the information terminating at those motor neurons are indeed going through the few core neurons, we need to also look at Figure 4.18. This plot shows the goodness of reduction for each motor neuron and we see a strong reduction goodness for most of the neurons.

Figure 4.19 shows how many core neurons would be needed to cover all paths terminating at each motor neuron. We ran our hourglass analysis with τ value of 100% to cover all paths of the network and got a large set of "core" neurons (91) to get the 100% coverage. Note although our original 10 core neurons were all interneurons, this extended set of core neurons has a lot of sensory and motor neurons. The figure shows a large number of motor neurons ($> 90\%$) being largely covered by the original 10 core neurons.

The right tail of Figure 4.17 and 4.18 has few neurons that completely bypass all the 10 core neurons. We looked at the neurons and found them to be mostly dual sensory-motor neurons, located at head regions and doing head muscle manipulation. The neurons were *URAD, IL1V, IL1D, IL1, URAV, DVB, RME*.

The right part of the distribution in Figure 4.19 shows around 30 neurons requiring a large number of "core" neurons to cover the paths going through them. The rightmost 10 neurons are the same neurons as previously mentioned that completely bypass the 10 core neurons and end up depending only on sensory neurons. The 20 neurons before that seem to depend on some neurons that could have made it to the core neurons if we had used a larger coverage threshold. This set of neurons are mostly motor (some motor-inter) neurons, related to head controlled locomotion. This set contains neurons like *RMDV, RMDD, RMD, SMDV, SMDD, RMF, RME, RMG, RIM, RIV*

4.8 Prior work in *C. elegans*

Varshney et al. [119] provides the most recent connectomic data for *C. elegans* that we also use. They analyzed the structural and topological properties of the connectome and found that several central neurons (based on closeness centrality) play a key role in information processing. Among them are command inter-neurons such as *AVA, AVB, AVE* which are responsible for locomotion control. On the other hand, *DVA, ADE,* etc. neurons have high out-closeness centrality and have a good position to spread a signal to the rest of the network. Most of their "central" neurons are in the core of our hourglass analysis and provide further evidence about their key role in information processing.

The existence of a set of highly connected and central hub vertices, termed as "rich-club", has been shown in the *C. elegans* connectome by Towlson et al. [115]. The set of our core neurons coincided almost entirely with the rich-club neurons, although the analysis was done with very different structural metrics and methods in the two cases. In addition to the important role of rich-club neurons in coordinated and adaptive movement, it was also shown that they were among the earliest developed neurons. This causes a strong connectivity among them.

A modular organization of the *C. elegans* connectome has been revealed by Sohn et al. [109] through cluster analysis. Their analysis reports that communities corresponded well to known functional circuits and it helps uncover the role of a few previously unknown neurons. They also identified a hierarchical organization among the five key clusters, which provide a backbone for higher-order complex behaviors. This is in agreement with our assumption of a strong underlying hierarchy for the flow of information controlling many of the organism's behavior.

The modular or mesoscale organization of the *C. elegans* connectome has been also analyzed by Pan et al. [87]. Their findings corroborate the notion of functional modularity where neurons in the same module tend to be physically located together and take part in a homogeneous activity. They contrasted this notion of modularity to that of one arising from wiring length minimization and communication cost efficiency and concluded that functional relevance is more important in shaping structure. Along these lines, they identified classes of critical neurons in every module, most of which have well known functionality including the command interneurons that also appear in our set of core neurons.

A study of the *C. elegans* connectome from developmental perspective by Varier and Kaiser [118] suggests that earlier born neurons tend to behave as network hubs. They have both longer and more connections than late-developed neurons. The neurons identified in our analysis as core are all early-developed ones and this also corroborates their centrality by having the advantage of creating connections early on.

A study about the optimality of the placement of neurons in *C. elegans* by Kaiser and Hilgetag [53] showed that the total wiring length can be reduces by 48% by optimally placing the neurons. However that would significantly increase the number of processing vertices along shortest paths between components as well. They concluded that neural systems are not exclusively optimized for global minimum wiring, but rather for a variety

of other factors of which the minimization of pair-wise processing steps is important. This study emphasizes the notion of choosing shorter communication path between neuron pairs and supports our choice of choosing paths that are shortest or near to shortest in terms of processing steps.

Whether neuronal placement of the *C. elegans* connectome minimizes total wiring cost was also examined by Chen et al. [24]. Their findings also conclude that the actual placement of neurons do not exactly correspond to globally minimized wiring. They did not have a clear explanation about the cause of this deviation but assumed there is an underlying tendency of clustering functionally related neurons.

The posterior nervous systems of the *C. elegans* was extensively analyzed by Jarrell et al. [52] to understand the decision making mechanism and behavior of this nematode. One of their key conclusions was that the nervous system mostly has a feedforward architecture that runs from sensory to motor neurons via interneurons. There is also some feedback circuitry in the nervous system and the actual physical output of the worm (i.e. motion etc.) feeds back to sensory neurons to allow closed control. There are however many feedforward loops (termed lateral connections in our analysis) that provide localized coordination most notably visible within interneurons. This is in alignment of our analysis of the dominant feedforward connections.

Analysis by Csoma et al. [30] challenged the well rooted notion of shortest path based communication routing in the human brain network. They collected empirical data through diffusion MRI and concluded that although a large number paths conform to the shortest path assumption, a significant fraction (20-40%) are inflated up to 4-5 hops. We took the conservative approach in our work and considered paths that are up to 2 hops larger than the shortest. It is difficult to obtain reliable empirical data about brain networks.

Research by Avena-Koenigsberger et al. [6, 7] analyzed in depth communication strate-

gies in the human brain and also challenged the shortest path assumption. They discussed how the computation of shortest path routing is not feasible in the brain circuitry, and the shortest path routes would leave out around 80% of the neural connections. They went through the spectrum of routing strategies hinging upon the amount of global information and communication required. At one end of the spectrum, there could be random walk based routing mechanism which is wasteful and fails short to achieve the desired outcome but requires no knowledge. On the other end is shortest path based routing requiring global wiring knowledge at each neuron. As a more realistic choice they studied the k-shortest path based approach (with k being 100). Their findings show that this strategy increase the utilization of connections. We have used a more relaxed constraint to chose paths between any two vertices by allowing all possible paths that are up to 2 hops larger than shortest path between the pair.

4.9 Summary

We have expanded the hourglass analysis framework into the area of general digraphs with the study of the *C. elegans* connectome in this chapter. Topologically the connectome network was highly cyclic with one large SCC containing more than 80% of the vertices. Our focus was to analyze this network following the feedforward flow of information that originates at the sensory neurons collecting outside information, gets integrated at interneurons and terminates at the motor neurons driving the organism's response. In order to identify the appropriate dependency paths in this network running from sensory to motor neurons and apply the hourglass framework, we analyzed most well known routing mechanism found in the brain network analysis literature. We found that the hourglass effect is present in the network to varying extent irrespective of the path routing strategy selected. The connectome was analyzed with respect to both the chemical synaptic network (which is more widely studied in the literature) and the gap/electric junction network. In both cases,

the network showed strong hourglass effect with nearly conserved set of "core" neurons identified.

The study of the identified "core" neurons of the hourglass waist revealed that they are among the most important "command interneruons" responsible for the organism's locomotion. Locomotion is a central task of this small organism that enables it to achieve most other required functionality. We reported well known functional circuits of the organism many of which contained the core neurons. We found the core neurons having a substantial overlap with rich-club neuron identified for the network with a very different topological analysis. We also observed that the core neurons were among the earliest developed neurons indicating their importance in the subsequent network connectivity structure.

An interesting observation was found while analyzing the hierarchical structure of this network. Utilizing the set of dependency paths, we proposed an hierarchy inference algorithm for digraphs. The outcome revealed a strong separation of hierarchies among the three neuron types of the network, but no hierarchy structure among almost all interneurons that contained our core neurons. This exemplified the situation when an hourglass effect be present even in the absence of any hierarchy within the network core.

Another key question we answered through analysis of the connectome is "what is the benefit of the hourglass architecture?". We explained the phenomenon of "dimensionality reduction" in which the input information is processed and compressed by a few number of intermediate units, which is later re-used and shared by many output units. This dimensionality reduction effectively reduce the dependency of the target units to the small core units instead of the large number of input units and provides flexibility and robustness in their functionality due to less connection overhead. For the connectome, we found that more than 90% of the motor neurons dependency can be entirely attributed to the 10 identified core neurons instead of the 88 sensory neurons demonstrating a strong dimensionality

reduction.

CHAPTER 5

KEY CONTRIBUTIONS, DISCUSSION AND POSSIBLE EXTENSIONS

5.1 Summary of key contributions

In summary, the main contributions of this thesis are:

1. To show how to transform a directed hierarchical network into a dependency network, and to introduce path centrality as an appropriate metric for the analysis of dependency networks.

2. To formulate the core identification problem as finding the smallest set of vertices that are traversed by a given fraction τ of all source-target paths.

3. To show how to quantify whether a dependency network exhibits the hourglass effect.

4. To propose a probabilistic "Reuse-Preference" model of dependency network formation, which illustrates the conditions under which a dependency network exhibits the hourglass effect.

5. To apply this analysis and modeling framework on several dependency networks from different disciplines, showing that they all exhibit the hourglass effect but to a varying extent and with different waist characteristics.

6. To compare the hourglass analysis framework with existing network "core finding" methods and compare path centrality with other vertex centrality metrics.

7. To extend the initial framework to general digraph networks that are not strictly hierarchical because they include feedback loops and lateral connections.

8. To apply the extended framework on the *C. elegans* brain network (connectome) and identify a core of ten neurons that almost all paths from sensory to motor neurons traverse.

9. To explain the role of core neurons in the *C. elegans* brain network as a dimensionality reduction mechanism, compressing the information provided by the 88 sensory neurons into a smaller set of intermediate-complexity functions that are re-used by the 119 motor neurons.

10. To propose a framework for inferring hierarchies in general digraphs utilizing dependency relationship information embedded in source-target paths.

11. To discuss the significance of the hourglass effect in both technology and nature in terms of network bottlenecks, cost, evolvability and robustness.

5.2 Discussion – significance of the hourglass effect

The hourglass effect is significant for several reasons. One of them is that the modules at the waist of the network create a "bottleneck" in the flow of information from sources (or inputs) to targets (or outputs). Such bottleneck network effects have been studied in the literature under different names. For instance, the term "core-periphery networks" has been broadly used in network science to refer to various static and dynamic topological properties (e.g., rich-club effect, onion-like networks) that result from a dense, cohesive core that is connected to sparsely connected peripheral vertices (but not necessarily organized in an acyclic input-output hierarchy) [15, 28, 99]. Bottlenecks have been also observed in gene regulatory networks [13], in protein networks [126], in general evolutionary models

[51], among many other domains. The methodology we have presented in this study for the identification of the core and for the quantification of the hourglass effect can serve as a unified approach for the study of bottleneck network phenomena in a wide range of disciplines.

Why do so many networks in nature and technology exhibit the hourglass effect? Is there a single underlying explanation or are there different mechanisms through which a hierarchical network can acquire this property? In technological networks, the reuse of existing modules has economic benefits in terms of design and implementation cost, and so it may be that the hourglass property results "by design" [124]. In natural networks, on the other hand, are there similar costs that an evolutionary process gradually reduces or should we look for a completely different explanation? The model of [37] captures how a realistic evolutionary process searches for the network that results in a desired input-output (linear) transformation. A more recent work [106] proposes an optimization-based framework, modeling sources as characters and targets as strings, that creates the given targets through the construction and reuse of intermediate substrings. The proposed RP-model offers a different, probably more general explanation for the hourglass effect: a dependency network with multiple sources and targets exhibits the hourglass effect when each vertex tends to depend on vertices of similar complexity (instead of connecting directly to sources or vertices of much lower complexity). This "preference for reuse" tends to create deep hierarchies in which a small set of intermediate vertices is traversed by most dependency paths. The RP-model is probabilistic, and so it is not possible to predict which specific intermediate vertices will emerge at the waist. In practice, we expect that the vertices at the waist will correspond to modules that are both highly general (meaning that their function is needed, directly or indirectly, by many targets) and highly complex (meaning that to provide that function, those modules need to utilize, directly or indirectly, the functionality of many sources).

The hourglass effect is also significant for the evolvability and robustness of hierarchically modular systems. Intuitively, the hourglass effect should allow a system to accommodate frequent changes in its sources or targets (i.e., to be able to evolve as the environment changes) because the few modules at the waist "decouple" the large number of sources from the large number of targets. If there is a change in the inputs (sources), the outputs do not need to be modified as long as the modules at the waist can still function properly. Similarly, if there is need for a new target, it may be much easier (or cheaper) to construct it reusing the modules at the waist rather than directly relying on sources. This is related to the notion of "constraints that de-constrain", introduced by Kirschner and Gerhart in the context of biological development and evolvability [63]. At the same time however, the presence of these critical modules at the waist (the "constraints") limit the space of all possible outputs that the system can generate ("phenotype space"), at least for a given maximum cost. The mechanisms through which the hourglass effect can improve evolvability but also limit the phenotype space is an important issue not only for natural systems but also for evolving technological systems [96].

Finally, understanding the implications of the hourglass effect for the cost, robustness, and evolvability of designed or technological systems can also have significant practical applications. In engineering, the primary focus is typically on optimality rather than on evolvability or robustness (e.g., design the minimum cost electronic circuit that can perform a given logic function). Such system-wide cost minimizations may appear attractive at first but they typically lead to non-hierarchical (monolithic) designs that are hard to test, evolve, or operate in the presence of failures. On the other hand, hierarchical design often lacks a systematic framework and the tools that would allow the designer to automatically identify, given a set of inputs and a set of outputs, the intermediate modules that would be most reusable. This becomes an even harder problem when we consider that most technological systems need to evolve as the desired functionalities (outputs) and conditions

(inputs) often change over time. One approach, which has not been pursued so far to the extent of our knowledge, is to start the design process from the waist, rather than bottom-up or top-down: first design a relatively small number of modules of intermediate complexity that will form the waist of the dependency network. Then, construct these modules based on the inputs, and in parallel construct the outputs based on these modules at the waist. Of course the key challenge in this approach is to develop algorithms and tools that can automatically identify those few central building blocks that will form the hourglass waist from the system specifications.

5.3 Possible extensions

5.3.1 Evolution of hourglass effect in software systems

A natural extension to hourglass framework analysis is to analyze the progression of the hourglass effect in an evolving system. Several prior studies have conducted artificial simulation of hourglass/bow-tie network evolution for layered network [3, 4, 37]. For the purpose of this study, we can leverage repositories of large open-source software systems that have been evolving over several years. Software systems as complex network have been studied extensively before [81, 116]. There are several open sourced software projects like Linux, OpenSSH, Samba, SQLite, Bind etc. with code availability.

Each of these open-sourced systems have two types of data:

1. Call graphs: A call graph shows the functions that are called by each function in the system. The call graph shows the architecture of the system, identifying the modules (functions) and their hierarchical relation (the dependencies of each function). For evolving systems, one challenge is to extract call graphs from the legacy systems.

The core Linux for example has been developed since 1991. Compiling the earlier versions to extract call graphs is challenging since current compiler can't handle old codes due to change in programming language specifications.

2. Software patches: This data repository shows which functions were modified in each version of the system, relative to the previous version. The magnitude of the changes (in terms of count lines of code) can also be extracted from the data.

We can study the amount and frequency of modification occurring to individual function and correlate them to their role in the hourglass structure.

5.3.2 The hourglass effect in cancer

Carcinogenesis and embryonic development have many similarities, despite their ultimately opposite outcome [61]. One major commonality between the two processes is the Epithelial-Mesenchymal Transition (EMT) through which epithelial cells, which are polarized and immotile, are transformed to motile, mesencymal cells with migratory protrusions [55]. In the case of development, the EMT process is activated at various stages, both in early embryogenesis (e.g., in neural crest formation) as well as in later stages (e.g., in cardiac valve formation in vertebrates). EMT is a reversible process. The opposite transition, from mesenchymal to epithelial cells (referred to as MET) is also common in development (e.g., in the formation of the kidneys epithelium).

In the case of cancer, EMT and MET play a crucial role in carcinoma progression and metastasis [21]. Most solid tumors in humans originate from mutated epithelial cells of different types that detach from their healthy neighboring cells and invade other cell layers and tissues. For this invasion to happen, the mutated cells have to acquire an appropriate shape and the capability to be highly motile. Given that cancers typically become untreat-

able or fatal when they reach the metastatic stage, it is not an exaggeration to say that EMT is the most critical step of the disease. If we had a way to somehow block EMT in cancerous cells, it is not unreasonable to imagine that we could slow down or even block carcinogenesis independent of the specific mutation(s) and tissues in which the cancerous cells first appeared. The so-called "developmental hourglass" is an old observation in biology but it has also come under new light in the last five years or so. Von Baer noticed in the early 19th century that there exists a certain developmental stage in which embryos of different animals look similar. Over the course of development, the youngest embryos of diverse species look very different, but progressively they converge towards a similar form, before they diverge again to achieve the tremendous diversity in adult forms. The stage where embryos of different animals look similar, or the waist of the hourglass, is referred to as the phylotypic stage.

With several recent publications in top-tier journals [31, 54], several intriguing findings re-instating the developmental hourglass have emerged. When the relative evolutionary age of genes expressed in different developmental stages of zebrafish was investigated, the genes expressed in the phylotypic stage were the most ancient [31]. Thus, genes shaping the most conserved phylotypic developmental stage are also evolutionarily the most conserved. Also, gene expression divergence across species, as well as gene expression noise within species, follows an hourglass pattern in fruitflies [54]. In other words, genes expressed in a mid-developmental stage, rather than in early or late stages, are under the strongest transcriptional constraint.

An earlier work from our lab developed a computational model that provides a plausible explanation for the emergence of the developmental hourglass [4]. The model considers a general hierarchical gene regulatory network that controls the developmental process, and the evolution of a population under random perturbations in the structure of that network. The model predicts, under fairly general assumptions, the emergence of an hourglass

pattern in the structure of a temporal representation of the underlying gene regulatory network. The evolutionary age of the corresponding genes also follows an hourglass pattern, with the oldest genes concentrated at the hourglass waist. The main condition behind the hourglass effect is that developmental regulators should have an increasingly specific function as development progresses. Analysis of developmental gene expression profiles from *Drosophila melanogaster* and *Arabidopsis thaliana* provided consistent results with our theoretical predictions.

Given the similarities between development and carcinogenesis, and given that the developmental process exhibits the hourglass effect, one initial hypothesis we can formulate is: Do the EMT and MET processes exhibit an hourglass effect during carcinogenesis? Suppose hypothetically that we knew the gene regulatory and signaling mechanisms that drive the EMT process. That would allow us to know how the EMT process unfolds over time. Then, we would say that the EMT process exhibits the hourglass effect if there is a certain temporal stage at which only a small number of regulatory genes are controlling the process, compared to the number of genes that participate at the onset of the process or at its later stages. Similarly, given the reversibility of EMT, we can make the same hypothesis about the MET process. The similarity between EMT and development is not the only reason we make the previous hypothesis. Another reason is the large diversity in mutational signatures observed even in cancers of the same type as well as the large phenotypic and functional heterogeneity of cancerous cells even in the same tumor. This suggests that carcinogenesis is a "multiple-input multiple-output" process. On the other hand, the high conservation of the EMT program and its generality across different cancers would probably not be possible if its core regulatory circuit involved many different genes and interactions. This mental picture of a system with many inputs and many outputs, controlled by a relatively small core that is general and highly conserved, is consistent with the hourglass hypothesis.

Appendices

APPENDIX A

A.1 Notation

Table A.1: List of symbols.

Symbol	Description
X	the cardinality of a set $\mathbf{X}$
$\mathbf{G_0}$	the original directed network (may include cycles)
$\mathbf{G}$	dependency network (directed and acyclic, by construction)
$\mathbf{V}$	set of vertices
$\mathbf{E}$	set of edges
$\mathbf{S}$	set of sources
$\mathbf{T}$	set of targets
$\mathbf{M}$	set of intermediates
$\mathbf{I}(v)$	set of vertices with edges to v – inputs of v
$\mathbf{O}(v)$	set of vertices with edges from v – outputs of v
$d_{in}(v)$	in-degree of v
$d_{out}(v)$	out-degree of v
$p(s,t)$	a path from a source s to a target t – an ST-path
$P(v)$	path centrality of v
$P_S(v)$	number of paths from sources to v (complexity of v)
$P_T(v)$	number of paths from v to targets (generality of v)
$\mathbf{P}$	set of all ST-paths
$\mathbf{P_R}$	set of ST-paths that traverse a set $\mathbf{R}$ of vertices
$\delta_{\mathbf{R}}$	path coverage of $\mathbf{R}(= P_R/P)$
$\hat{\mathbf{R}}_k$	set of k vertices with maximum path coverage
$\hat{\delta}_k$	path coverage of $\hat{\mathbf{R}}_k$
τ	path coverage threshold
$\mathbf{C}(\tau)$	a core (there may be more than one) for a given path coverage threshold τ
$\delta_{\mathbf{C}(\tau)}$	the path coverage of $\mathbf{C}(\tau)$ (may be more than τ)
$\mathbf{G_f}$	the flat network that corresponds to $\mathbf{G}$
$\mathbf{C_f}(\tau)$	the core of the flat network $\mathbf{G_f}$ for the threshold τ
$H(\tau)$	H-score for the threshold τ
$U_{\mathbf{C}}$	core vertex coverage of core $\mathbf{C}$
$\mathbf{V_{ST}}$	set of vertices in at least one ST-path
$\phi_{\mathbf{C}}(v)$	indicator variable that vertex $v \in \mathbf{V_{ST}}$ is reachable from, or can reach, at least one vertex in core $\mathbf{C}$
$L(v)$	location of vertex v
$L_{\mathbf{C}}$	average location of core $\mathbf{C}$
$\delta_{\mathbf{C}}(v)$	incremental path coverage when vertex v is added in core $\mathbf{C}$ (also referred to as the "weight" of core vertex v)
α	reuse preference exponent
d_{in}	average in-degree
β	parameter of edge-copying model

A.2 Submodularity of the C^{3}MC objective function

Lemma A.2.1. *The objective function of the C^{3}MC problem is submodular, i.e.,*

$$\delta_{\mathbf{X}\cup\{v\}} - \delta_{\mathbf{X}} \geq \delta_{\mathbf{Y}\cup\{v\}} - \delta_{\mathbf{Y}} \tag{A.1}$$

for any $\mathbf{X}, \mathbf{Y}$ *such that* $\mathbf{X} \subseteq \mathbf{Y} \subseteq \mathbf{V}$ *and for any vertex* v *in* $\mathbf{V}$.

Proof. The function $\delta_{\mathbf{R}}$ is non-negative and non-decreasing.

- Case 1: Consider all ST-paths that traverse v but not any vertex in $\mathbf{Y}$. These paths do not traverse any vertex in $\mathbf{X}$ either. So the increase in the coverage of $\mathbf{X}$ and $\mathbf{Y}$ will be the same when we add v in both sets.
- Case 2: Consider all ST-paths that traverse v as well as one or more vertices of $\mathbf{Y}$ but not any vertex of $\mathbf{X}$. Such ST-paths increase the coverage of only $\mathbf{X} \cup v$.
- Case 3: Consider all ST-paths that traverse v as well as one or more vertices of $\mathbf{X}$. These paths are already included in the coverage of both $\mathbf{X}$ and $\mathbf{Y}$, and so they will not cause any further coverage increase by including v in $\mathbf{X}$ and $\mathbf{Y}$.

These three cases account for all ST-paths traversing v and we have shown that the submodularity condition is satisfied in all of them. □

A.3 NP-Completeness proof of the τ-Core problem

Theorem A.3.1. *The τ-Core problem is NP-Complete.*

Proof. It is easy to see that τ-Core $\in NP$ since we can compute the path coverage of a set of vertices in polynomial time.

In a DAG G, vertices with zero in-degree are called sources, and vertices with zero out-degree are called targets. The k-GCM(#P) problem asks for a set of vertices of cardinality $\leq k$ in G that maximize the number of covered source-to-target (ST) paths. This problem was shown to be NP-Complete in [50]. The τ-Core problem asks for a minimum-cardinality set of vertices in G that covers at least a fraction τ of all possible ST paths. We proceed by reducing the k-GCM(#P) problem to the τ-Core problem.

Define a *complete-chain* as a DAG that has one source and one target, with each pair of vertices connected by one directed edge. Such a DAG of n vertices has 2^{n-2} ST paths, which is the maximum number of ST paths for any DAG of that size (see lemma A.3.2). So, the number of ST paths in a simple DAG (i.e. without multi-edges) can grow at most exponentially with the number of vertices.

We first invoke the τ-Core solver with $\tau = 1$. If the solutions cardinality $\leq k$, we have found a solution for the k-GCM(#P) problem. Otherwise, we initiate a binary search over all the possible discrete values of τ that correspond to the finite number of ST paths in the network. We utilize the τ-Core solver to find the largest τ for which a solution set with cardinality k exists. Any larger value of τ would require at least $(k+1)$ vertices and hence the returned τ is a solution for the k-GCM(#P) problem. Given that the search space can be exponentially large (2^{n-2} in the worst case), a binary search will require at most a linear number of invocations of the τ-Core solver. Therefore k-GCM(#P) $\leq_p$ τ-Core. □

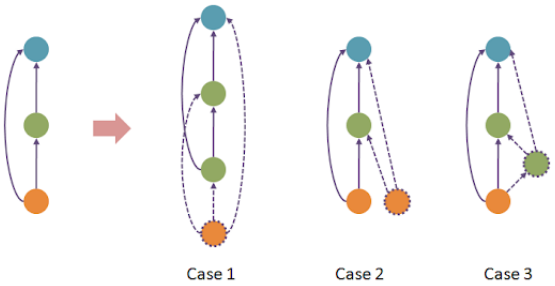


Figure A.1: The leftmost graph is a complete-chain DAG of 3 vertices. In the right graph we show three ways to build a DAG of 4 vertices. Newly added vertices and edges are shown in dotted format.

Lemma A.3.2. *The complete-chain DAG of n vertices has the maximum number of ST paths among all DAGs with the same size.*

Proof. It is easy to see that the complete-chain DAG of n vertices has 2^{n-2} ST paths. It is also easy to see that if we add one more vertex in a complete-chain DAG, the number of ST paths doubles. We show next that no other DAG of n vertices can have more than 2^{n-2} ST paths.

For DAGs having less than two vertices we do not have ST paths. For a DAG of two vertices, only a single chain DAG can be formed and the number of ST paths is one. Let us assume our claim is true for a DAG of n vertices. We will show inductively that our claim is also true for any DAG of $(n+1)$ vertices.

Starting from a complete-chain DAG of n vertices, we have three ways (shown in Figure A.1) to add a new vertex to the DAG and form another DAG with the maximum number of edges.

1. Add a vertex as the new single source or target creating a complete-chain DAG of $(n+1)$ vertices, which trivially satisfies our claim.

2. Add a vertex as a new second source or target. This vertex will create a number of

new paths, equal to the number of paths the original source/target has. Hence this also doubles the number of ST paths.

3. Add a vertex as a new intermediate vertex. This will again create a complete-chain DAG of $(n+1)$ vertices, sasisfying our claim.

These three cases complete our previous claim for a DAG of $(n+1)$ vertices, and hence by induction prove the lemma. □

APPENDIX B

B.1 Vertices at the waist of each dependency network

Table B.1: The waist of the OpenSSH-v5.2 call-graph network.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
packet_send	0.50	0.50	*Wrapper function for formatting and sending TCP packet.*
packet_read_seqnr	0.37	0.20	*Function to return type of received packet.*
do_exec	0.42	0.10	*Function responsible for spawning a sub-shell as part of session creation*

Table B.2: The waist of the Apache-Math-v3.4 call-graph network. SCCs are listed in Table B.3.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
SCC-1	0.60	0.60	*Methods from the decimal floating point library class.*
Vector3D:init	0.08	0.06	*Initializer for base class implementing vectors in a three-dimensional space.*
DerivativeStructure:init	0.10	0.04	*Initializer for base class that is the workhorse of differentiation library.*
FastMath:abs	0.04	0.03	*Faster math library's absolute value computing method.*
EigenDecomposition:init	0.10	0.02	*Initializer for the class handling eigen decomposition of a real matrix.*
BigFraction:init	0.05	0.02	*Initializer for base class representing a rational number without any overflow.*
MatrixUtils:createRealMatrix	0.09	0.01	*Method to create and initialize a real-valued matrix from given data.*
Line:init	0.04	0.01	*Initializer for the three dimensional geometric line Java class.*
IntervalsSet:iterator	0.008	0.01	*Iterator for traversing a set of one dimensional geometric intervals.*

Table B.3: SCCs in the core of the Apache-Math-v3.4 call-graph network.

SCC	Components
SCC-1	*DfpMath:splitPow, Dfp:lessThan, Dfp:align, DfpMath:split, Dfp:dotrap, Dfp:multiply, Dfp:divide, DfpMath:log, Dfp:multiplyFast, DfpMath:logInternal, Dfp:negate, Dfp:add, Dfp:remainder, Dfp:init, Dfp:power10K, Dfp:trunc, Dfp:unequal, DfpMath:splitMult, Dfp:subtract, DfpMath:exp, Dfp:floor, Dfp:newInstance, DfpField:newDfp, Dfp:toDouble, Dfp:rint, Dfp:greaterThan, Dfp:round, DfpMath:pow, DfpMath:expInternal, Dfp:intValue, Dfp:copysign*

Table B.4: The waist of the Rat (*R. Norvegius*) metabolic network. SCCs are listed in Table B.5.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
SCC-1	0.60	0.60	*Contains the metabolic precursors: Pyruvate, PhosphenolPyruvate, Oxalocetate.*
Arachidonate	0.08	0.09	*Essential for enzyme synthesis.*
Acetyl-CoA	0.25	0.05	*A metabolic precursor.*
SCC-2	0.25	0.04	*Contains the metabolic precursors: Glycerone Phosphate, Ribose-5-Phosphate, Glyceraldehyde 3 Phosphate.*
Phosphatidate	0.07	0.04	*Essential for Lipid synthesis.*
SCC-3	0.02	0.02	*These compounds take part in Purine metabolism.*
{GQ1b, Glycan 9-11}	0.01	0.01	*These compounds take part in Ganglioside metabolism.*

Table B.5: SCCs in the core of the Rat (*R. Norvegius*) metabolic network.

SCC	Components
SCC-1	*Ammonia, Pyruvate, Oxalocetate, L-Alanine, L-Aspartate, Glutathione, Glycine, L-Arginine, L-Glutamine, L-Serine, Phosphoenolpyruvate, gamma-Glutamylcysteine, L-Argininosuccinate, Mercaptopyruvate, L-Cystathionine, Casbene, Carbomoyl Phosphate, Fumarate, Citrolline, Malate, L-Cysteine, L-Ornithine*
SCC-2	*Glycerone Phosphate, Ribose 5-Phosphate, Glyceraldehyde 3-Phosphate, PRPP, D-Xylulose 5-Phosphate, beta-D-Fructose-6-phosphate, Sedoheptulose 7-phosphate, beta-D-Fructose 1,6-bisphosphate*
SCC-3	*AMP, GDP, DNA, Guanine, Deoxyadenosine, dATP, IMP, dGTP, XMP, Xanthine, Guanosine, dADP, dAMP, dGDP, Inosine, Adenine, Hypoxanthine*

Table B.6: The waist of the Monkey (*M. Mulatta*) metabolic network. SCCs are listed in Table B.7.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
SCC-1	0.57	0.57	*Contains the metabolic precursors Pyruvate, PhosphenolPyruvate, Oxalocetate.*
Arachidonate	0.10	0.09	*Essential for enzyme synthesis.*
Acetyl-CoA	0.25	0.06	*A metabolic precursor.*
Phosphatidate	0.08	0.05	*Essential for lipid synthesis.*
SCC-2	0.21	0.03	*Contains the metabolic precursors: Glycerone Phosphate, Ribose-5-Phosphate, Glyceraldehyde 3 Phosphate.*
SCC-3	0.03	0.03	*These compounds take part in Purine metabolism.*
Lc3Cer	0.01	0.01	*Aids in biosynthesis of Glycolipids.*
Malonyl-[acp]	0.01	0.01	*A key compound for fatty acid synthesis.*

Table B.7: SCCs in the core of the Monkey (*M. Mulatta*) metabolic network.

SCC	Components
SCC-1	*Ammonia, Pyruvate, Oxalocetate, L-Alanine, L-Aspartate, Glutathione, Glycine, L-Arginine, L-Glutamine, L-Serine, Phosphoenolpyruvate, gamma-Glutamylcysteine, L-Argininosuccinate, Mercaptopyruvate, Cyc-Gly, Carbomoyl Phosphate, Fumarate, Citrolline, Malate, L-Cysteine, L-Ornithine*
SCC-2	*Glycerone Phosphate, Ribose 5-Phosphate, Glyceraldehyde 3-Phosphate, PRPP, D-Xylulose 5-Phosphate, beta-D-Fructose-6-phosphate, Sedoheptulose 7-phosphate, beta-D-Fructose 1,6-bisphosphate*
SCC-3	*AMP, GDP, DNA, Guanine, Deoxyadenosine, dATP, IMP, dGTP, XMP, Xanthine, Guanosine, dADP, dAMP, dGDP, Inosine, Adenine, Hypoxanthine, Adenosine, GMP, Adenylosuccinate, Xanthosine*

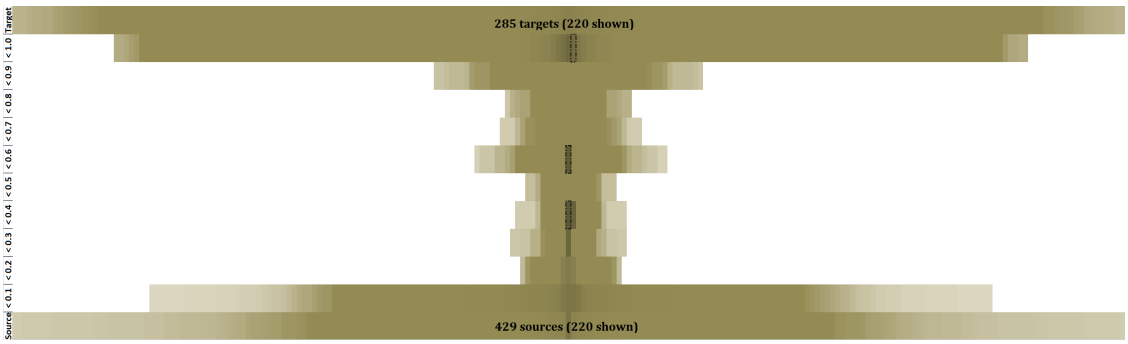
Table B.8: The waist of the SCotUS citation network on Abortion cases. Cases labeled as "landmarks" are listed as Historic by the Legal Information Institute at Cornell University.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
Planned Parenthood v. Casey (1992)	0.69	0.69	*A "landmark" decision on abortion rights.*
Roe v. Wade (1973)	0.65	0.20	*A "landmark" decision in favor of abortion rights with certain restrictions.*
Bigelow v. Virginia (1975)	0.38	0.05	*A "landmark" decision on protecting First Amendment right on advertising, where the advertisement in question was on abortion services.*
Harris v. McRae (1980)	0.55	0.03	*A "landmark" decision regarding federal funds restriction on abortion.*

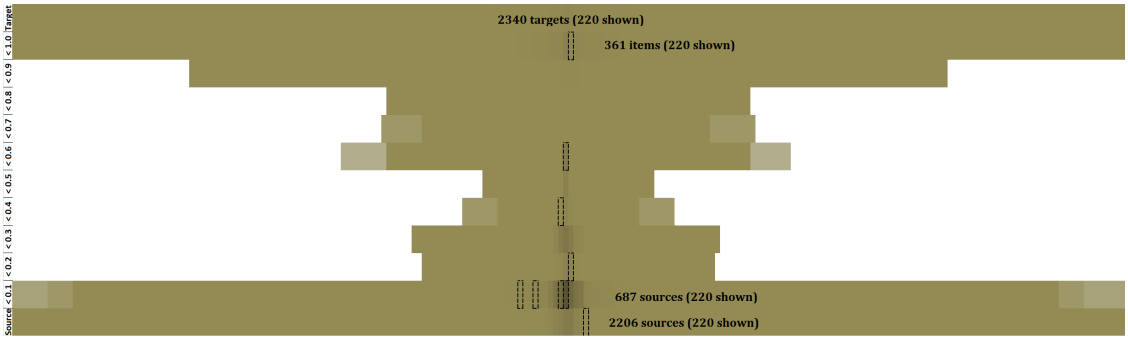
Table B.9: The waist of the SCotUS citation network on Pension cases. Cases labeled as "landmarks" are listed as Historic by the Legal Information Institute at Cornell University.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
Goldberg v. Kelly (1970)	0.42	0.42	*A "landmark" decision that established the full evidential hearing requirement before termination of welfare benefits.*
Allied Structural Steel Co. v. Spannaus (1978)	0.22	0.22	*A "landmark" decision that reinstated pension rights for certain Allied Steel employees.*
L.A. Dept. of Water & Power v. Manhart (1978)	0.16	0.11	*A "landmark" decision that stated discrimination in pension contribution requirement based on sex is unlawful.*
US Railroad Retirement Bd. v. Fritz (1980)	0.38	0.09	*A "landmark" decision that reinstated pension rights for certain US Railroad employees.*
Johnson v. Robison (1974)	0.21	0.03	*A decision that retained certain benefits for combat veterans.*
Hishon v. King & Spalding (1984)	0.08	0.02	*A decision regarding benefit discrimination based on sex.*
Helvering v. Davis (1937)	0.06	0.02	*A "landmark" decision defending the constitutional validity of the Social Security Act.*
Nollan v. California Coastal Com. (1987)	0.07	0.01	*A decision concerning 5th and 14th amendment for property protection.*
United States v. Kokinda (1990)	0.11	0.01	*A decision involving first amendment rights for free speech.*
Pension Benefit Guar. Corp. v. LTV Corp. (1990)	0.17	0.01	*A decision involving insurance of pension benefits.*
Plaut v. Spendthrift Farm (1995)	0.11	0.01	*A decision concerning separation of power between legislation and judiciary.*

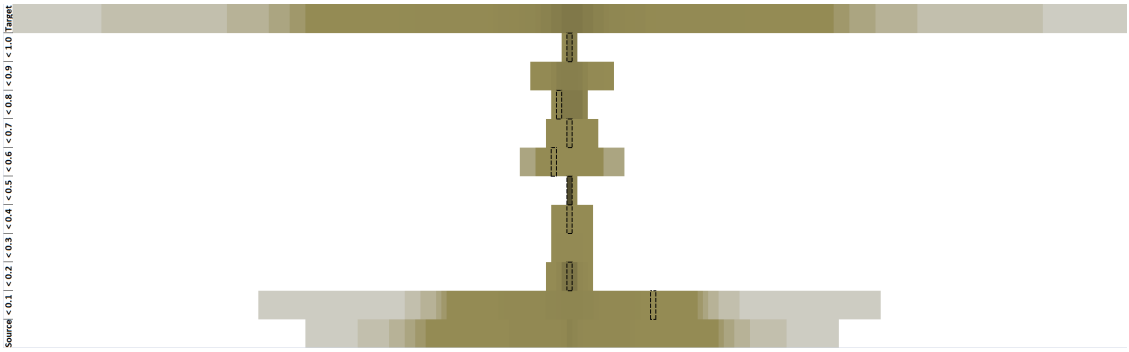
B.2 Location metric visualization for real networks



(a) OpenSSH call-graph

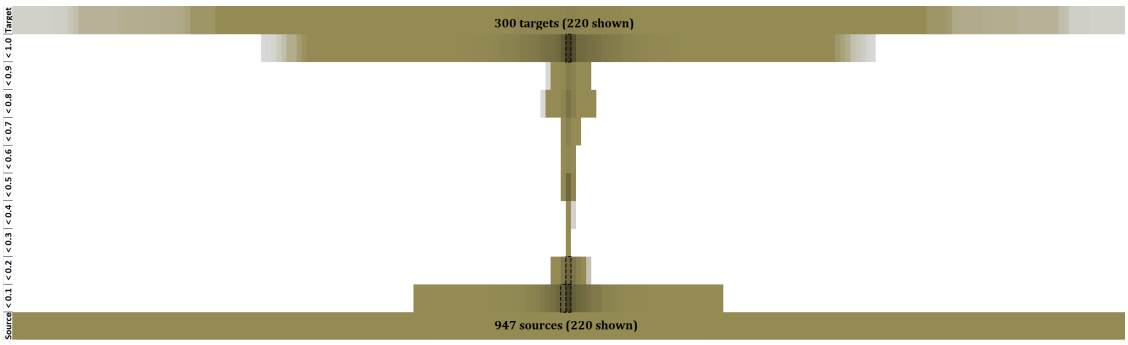


(b) Apache math call-graph

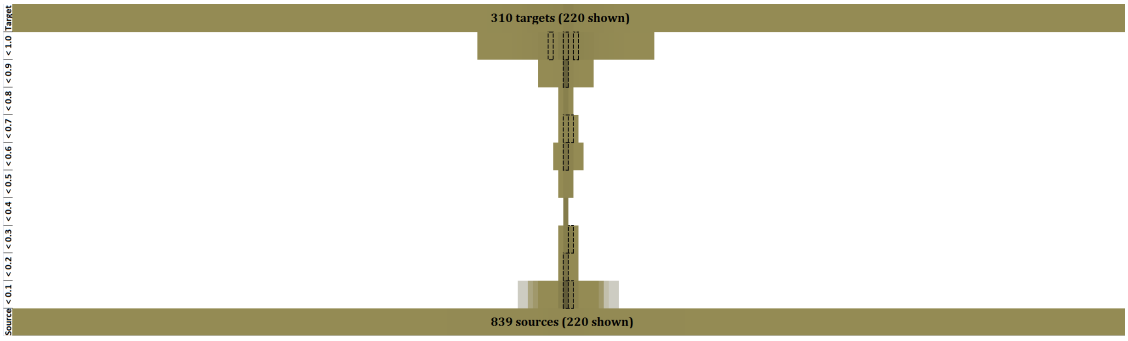


(c) Monkey metabolic network

Figure B.1: Visualizations of the location and path centrality for each network. Please refer to the caption of Figure 2.7 for a description of this visualization.



(a) Abortion case citation network



(b) Pension case citation network

Figure B.1: Continued.

B.3 Comparing RP-model generated synthetic network with real networks

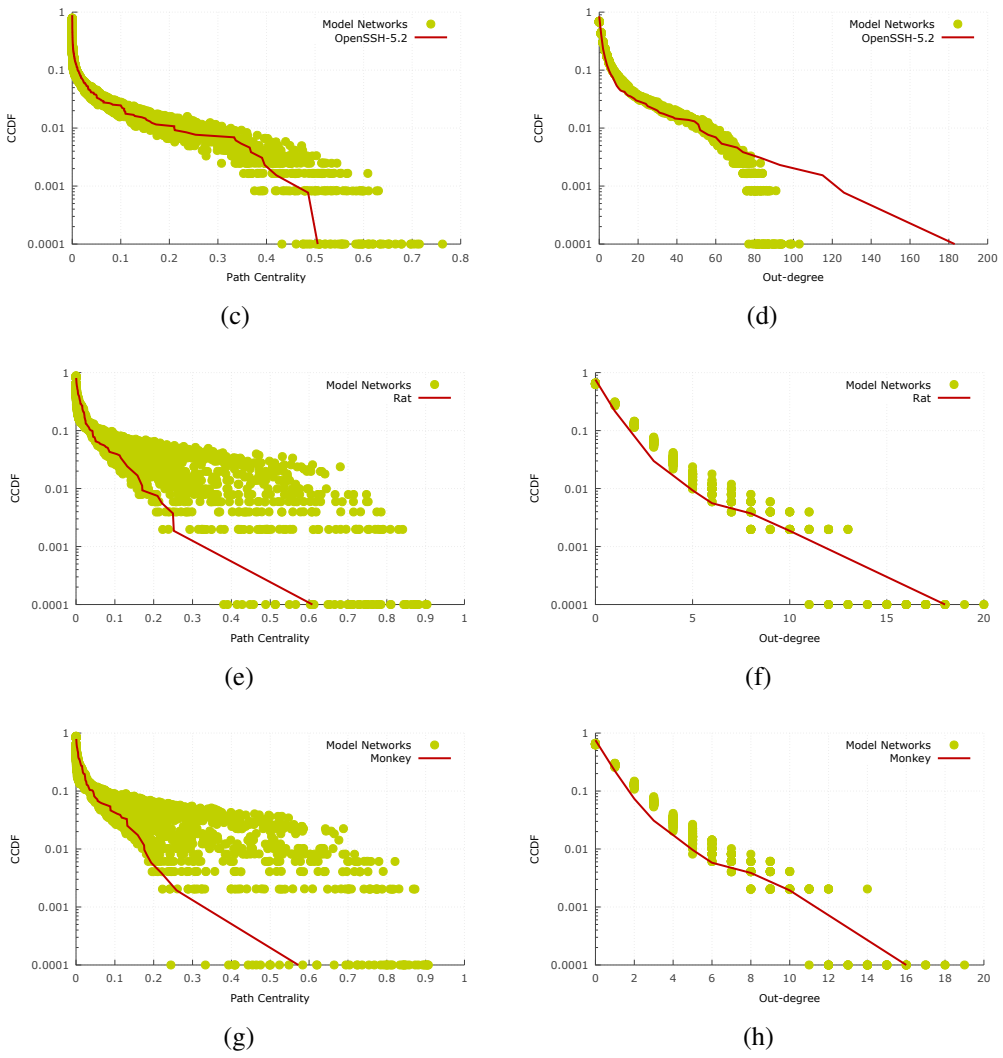


Figure B.2: Comparison of path centrality and out-degree distributions between some real dependency networks and the corresponding synthetic networks generated by the RP-model.

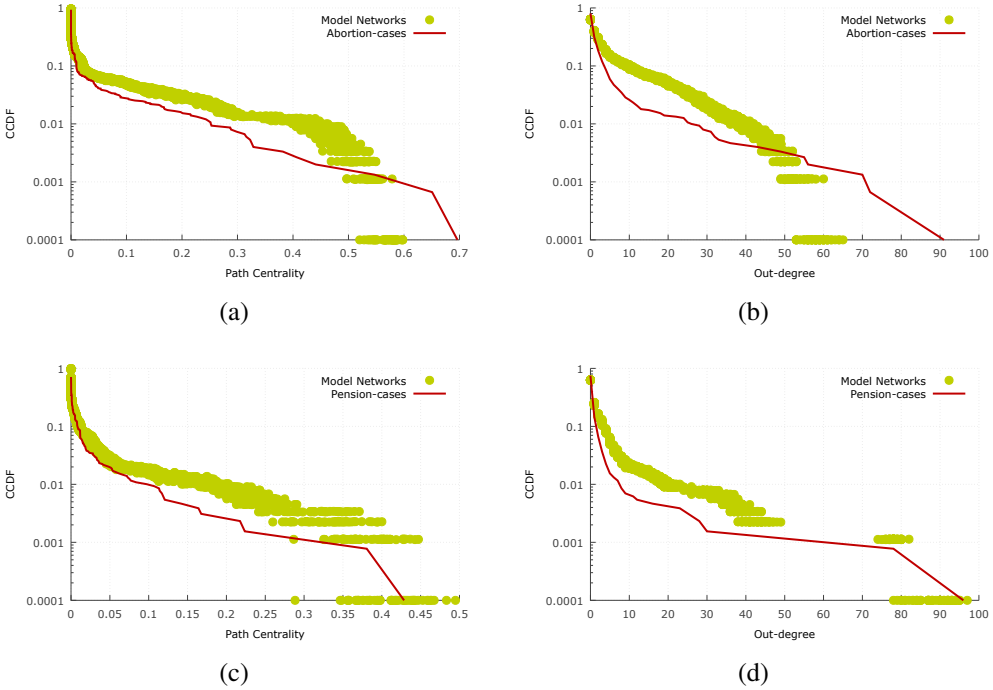


Figure B.2: Continued.

REFERENCES

[1] F. Abdelnour, H. U. Voss, and A. Raj, "Network diffusion accurately models the relationship between structural and functional brain connectivity networks," *Neuroimage*, vol. 90, pp. 335–347, 2014.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Prentice hall, 1993.

[3] S. Akhshabi and C. Dovrolis, "The evolution of layered protocol stacks leads to an hourglass-shaped architecture," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 206–217, 2011.

[4] S. Akhshabi, S. Sarda, C. Dovrolis, and S. Yi, "An explanatory evo-devo model for the developmental hourglass," *f1000research*, vol. 3, 2014.

[5] B Alberts, A Johnson, J Lewis, M Raff, K Roberts, and P Walter, *Molecular Biology of the Cell*, 4th. Garland Science, 2002.

[6] A. Avena-Koenigsberger, B. Mišić, R. X. Hawkins, A. Griffa, P. Hagmann, J. Goñi, and O. Sporns, "Path ensembles and a tradeoff between communication efficiency and resilience in the human connectome," *Brain Structure and Function*, vol. 222, no. 1, pp. 603–618, 2017.

[7] A. Avena-Koenigsberger, B. Misic, and O. Sporns, "Communication dynamics in complex brain networks," *Nature Reviews Neuroscience*, vol. 19, no. 1, p. 17, 2018.

[8] L Avery and J. H. Thomas, "Feeding and defecation," 1997.

[9] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity*. MIT press, Cambridge, 2000, vol. 1.

[10] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[11] C. I. Bargmann, "Chemosensation in c. elegans," 2006.

[12] B. Beutler, "Inferences, questions and possibilities in toll-like receptor signalling," *Nature*, vol. 430, no. 6996, pp. 257–263, 2004.

[13] N. Bhardwaj, K.-K. Yan, and M. B. Gerstein, "Analysis of diverse regulatory networks in a hierarchical context shows consistent tendencies for collaboration in the middle levels," *Proceedings of the National Academy of Sciences*, vol. 107, no. 15, pp. 6841–6846, 2010.

[14] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *Proceedings of the 2012 International Conference on Software Engineering*, IEEE Press, 2012, pp. 419–429.

[15] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social networks*, vol. 21, no. 4, pp. 375–395, 2000.

[16] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, no. 1, pp. 309–320, 2000.

[17] E. Bullmore and O. Sporns, "The economy of brain network organization," *Nature Reviews Neuroscience*, vol. 13, no. 5, p. 336, 2012.

[18] W. Callebaut and D. Rasskin-Gutman, *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. MIT press, Cambridge, 2005.

[19] A. Capocci, V. D. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, and G. Caldarelli, "Preferential attachment in the growth of social networks: The Internet encyclopedia Wikipedia," *Physical Review E*, vol. 74, no. 3, p. 036 116, 2006.

[20] T. Casci, "Development: Hourglass theory gets molecular approval," *Nature Reviews Genetics*, vol. 12, no. 2, pp. 76–76, 2011.

[21] C. L. Chaffer and R. A. Weinberg, "A perspective on cancer cell metastasis," *Science*, vol. 331, no. 6024, pp. 1559–1564, 2011.

[22] S. H. Chalasani, N. Chronis, M. Tsunozaki, J. M. Gray, D. Ramot, M. B. Goodman, and C. I. Bargmann, "Dissecting a circuit for olfactory behaviour in caenorhabditis elegans," *Nature*, vol. 450, no. 7166, p. 63, 2007.

[23] M. Chalfie, J. E. Sulston, J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, "The neural circuit for touch sensitivity in caenorhabditis elegans," *Journal of Neuroscience*, vol. 5, no. 4, pp. 956–964, 1985.

[24] B. L. Chen, D. H. Hall, and D. B. Chklovskii, "Wiring optimization can relate neuronal structure and function," *Proceedings of the National Academy of Sciences*, vol. 103, no. 12, pp. 4723–4728, 2006.

[25] J. Clune, J.-B. Mouret, and H. Lipson, "The evolutionary origins of modularity," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 280, no. 1755, p. 20 122 863, 2013.

[26] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani, "Detecting rich-club ordering in complex networks," *Nature physics*, vol. 2, no. 2, p. 110, 2006.

[27] B. Corominas-Murtra, J. Goñi, R. V. Solé, and C. Rodríguez-Caso, "On the origins of hierarchy in complex networks," *Proceedings of the National Academy of Sciences*, vol. 110, no. 33, pp. 13 316–13 321, 2013.

[28] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, "Structure and dynamics of core/periphery networks," *Journal of Complex Networks*, vol. 1, no. 2, pp. 93–123, 2013.

[29] M. Csete and J. Doyle, "Bow ties, metabolism and disease," *TRENDS in Biotechnology*, vol. 22, no. 9, pp. 446–450, 2004.

[30] A. Csoma, A. Kőrösi, G. Rétvári, Z. Heszberger, J. Bíró, M. Slíz, A. Avena-Koenigsberger, A. Griffa, P. Hagmann, and A. Gulyás, "Routes obey hierarchy in complex networks," *Scientific reports*, vol. 7, no. 1, p. 7243, 2017.

[31] T. Domazet-Lošo and D. Tautz, "A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns," *Nature*, vol. 468, no. 7325, pp. 815–818, 2010.

[32] M Driscoll and J Kaplan, "Mechanotransduction," 1997.

[33] D. Easley and J. Kleinberg, *Networks, crowds, and markets*, 2010.

[34] M. A. Fortuna, J. A. Bonachela, and S. A. Levin, "Evolution of a modular software network," *Proceedings of the National Academy of Sciences*, vol. 108, no. 50, pp. 19 985–19 989, 2011.

[35] J. H. Fowler and S. Jeon, "The authority of supreme court precedent," *Social Networks*, vol. 30, no. 1, pp. 16–30, 2008.

[36] J. H. Fowler, T. R. Johnson, J. F. Spriggs, S. Jeon, and P. J. Wahlbeck, "Network analysis and the law: Measuring the legal importance of precedents at the us supreme court," *Political Analysis*, vol. 15, no. 3, pp. 324–346, 2007.

[37] T. Friedlander, A. E. Mayo, T. Tlusty, and U. Alon, "Evolution of bow-tie architectures in biology," *PLoS Comput Biol*, vol. 11, no. 3, e1004055, 2015.

[38] P. A. Garrity, M. B. Goodman, A. D. Samuel, and P. Sengupta, "Running hot and cold: Behavioral strategies, neural circuits, and the molecular machinery for

thermotaxis in c. elegans and drosophila," *Genes & development*, vol. 24, no. 21, pp. 2365–2382, 2010.

[39] M. B. Goodman, "Mechanosensation.," *WormBook: the online review of C. elegans biology*, pp. 1–14, 2006.

[40] M. Gorman, *Codeviz: A callgraph visualiser*, `http://www.csn.ul.ie/~mel/projects/codeviz/`, 2015.

[41] G. Gousios, *Java-callgraph: Java call graph utilities*, `https://github.com/gousiosg/java-callgraph`, 2015.

[42] M. Granovetter, "Threshold models of collective behavior," *American journal of sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.

[43] J. M. Gray, J. J. Hill, and C. I. Bargmann, "A circuit for navigation in caenorhabditis elegans," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 9, pp. 3184–3191, 2005.

[44] Z. V. Guo, A. C. Hart, and S. Ramanathan, "Optical interrogation of neural circuits in caenorhabditis elegans," *Nature methods*, vol. 6, no. 12, p. 891, 2009.

[45] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[46] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[47] P. Holme, "Core-periphery organization of complex networks," *Physical Review E*, vol. 72, no. 4, p. 046 111, 2005.

[48] C.-C. Huang and A. Kusiak, "Modularity in design of products and systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 1, pp. 66–77, 1998.

[49] S. J. Husson, A. Gottschalk, and A. M. Leifer, "Optogenetic manipulation of neural activity in c. elegans: From synapse to circuits and behaviour," *Biology of the Cell*, vol. 105, no. 6, pp. 235–250, 2013.

[50] V. Ishakian, D. Erdös, E. Terzi, and A. Bestavros, "A framework for the evaluation and management of network centrality.," in *SDM*, SIAM, 2012, pp. 427–438.

[51] S. Jain and S. Krishna, "Large extinctions in an evolutionary model: The role of innovation and keystone species," *Proceedings of the National Academy of Sciences*, vol. 99, no. 4, pp. 2055–2060, 2002.

[52] T. A. Jarrell, Y. Wang, A. E. Bloniarz, C. A. Brittin, M. Xu, J. N. Thomson, D. G. Albertson, D. H. Hall, and S. W. Emmons, "The connectome of a decision-making neural network," *Science*, vol. 337, no. 6093, pp. 437–444, 2012.

[53] M. Kaiser and C. C. Hilgetag, "Nonoptimal component placement, but short processing paths, due to long-distance projections in neural systems," *PLoS computational biology*, vol. 2, no. 7, e95, 2006.

[54] A. T. Kalinka, K. M. Varga, D. T. Gerrard, S. Preibisch, D. L. Corcoran, J. Jarrells, U. Ohler, C. M. Bergman, and P. Tomancak, "Gene expression divergence recapitulates the developmental hourglass model," *Nature*, vol. 468, no. 7325, p. 811, 2010.

[55] R. Kalluri and R. A. Weinberg, "The basics of epithelial-mesenchymal transition," *The Journal of clinical investigation*, vol. 119, no. 6, pp. 1420–1428, 2009.

[56] M. Kanehisa and S. Goto, "Kegg: Kyoto encyclopedia of genes and genomes," *Nucleic Acids Research*, vol. 28, no. 1, pp. 27–30, 2000.

[57] M. Kanehisa, S. Goto, Y. Sato, M. Kawashima, M. Furumichi, and M. Tanabe, "Data, information, knowledge and principle: Back to metabolism in kegg," *Nucleic Acids Research*, vol. 42, no. D1, pp. D199–D205, 2014.

[58] B. Karrer and M. E. Newman, "Random graph models for directed acyclic networks," *Physical Review E*, vol. 80, no. 4, p. 046 110, 2009.

[59] N. Kashtan and U. Alon, "Spontaneous evolution of modularity and network motifs," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 39, pp. 13 773–13 778, 2005.

[60] N. Kashtan, E. Noor, and U. Alon, "Varying environments can speed up evolution," *Proceedings of the National Academy of Sciences*, vol. 104, no. 34, pp. 13 711–13 716, 2007.

[61] F. C. Kelleher, D. Fennelly, and M. Rafferty, "Common critical pathways in embryogenesis and cancer," *Acta oncologica*, vol. 45, no. 4, pp. 375–388, 2006.

[62] T. Kimata, H. Sasakura, N. Ohnishi, N. Nishio, and I. Mori, "Thermotaxis of c. elegans as a model for temperature perception, neural information processing and neural plasticity," in *Worm*, Taylor & Francis, vol. 1, 2012, pp. 31–41.

[63] M. Kirschner and J. Gerhart, "Evolvability," *Proceedings of the National Academy of Sciences*, vol. 95, no. 15, pp. 8420–8427, 1998.

[64] H Kirsten and P. Hogeweg, "Evolution of networks for body plan patterning; interplay of modularity, robustness and evolvability," *PLoS Comput Biol*, vol. 7, no. 10, e1002208, 2011.

[65] H. Kitano, "Biological robustness," *Nature Reviews Genetics*, vol. 5, no. 11, pp. 826–837, 2004.

[66] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins, "The web as a graph: Measurements, models, and methods," pp. 1–17, 1999.

[67] S. Kojaku and N. Masuda, "Finding multiple core-periphery pairs in networks," *Physical Review E*, vol. 96, no. 5, p. 052 313, 2017.

[68] P. L. Krapivsky and S. Redner, "Network growth by copying," *Physical Review E*, vol. 71, no. 3, p. 036 118, 2005.

[69] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins, and E. Upfal, "The web as a graph," in *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM, 2000, pp. 1–10.

[70] M. Lechner, R. Grosu, and R. M. Hasani, "Worm-level control through search-based reinforcement learning," *arXiv preprint arXiv:1711.03467*, 2017.

[71] Legal Information Institute - Cornell University Law School, *Historic Supreme Court Decisions*, `https://www.law.cornell.edu/`, Accessed: 2015-10-30, 2015.

[72] W. Li, Z. Feng, P. W. Sternberg, and X. S. Xu, "A c. elegans stretch receptor neuron revealed by a mechanosensitive trp channel homologue," *Nature*, vol. 440, no. 7084, p. 684, 2006.

[73] S. R. Lockery, "Neuroscience: A social hub for worms," *Nature*, vol. 458, no. 7242, p. 1124, 2009.

[74] D. M. Lorenz, A. Jeng, and M. W. Deem, "The emergence of modularity in biological systems," *Physics of Life Reviews*, vol. 8, no. 2, pp. 129–160, 2011.

[75] H.-W. Ma and A.-P. Zeng, "The connectivity structure, giant strong component and centrality of metabolic networks," *Bioinformatics*, vol. 19, no. 11, pp. 1423–1430, 2003.

[76] E. Z. Macosko, N. Pokala, E. H. Feinberg, S. H. Chalasani, R. A. Butcher, J. Clardy, and C. I. Bargmann, "A hub-and-spoke circuit drives pheromone attraction and social behaviour in c. elegans," *Nature*, vol. 458, no. 7242, p. 1171, 2009.

[77] H. Mengistu, J. Huizinga, J.-B. Mouret, and J. Clune, "The evolutionary origins of hierarchy," *PLoS computational biology*, vol. 12, no. 6, e1004829, 2016.

[78] D. Meunier, R. Lambiotte, and E. T. Bullmore, "Modular and hierarchically modular organization of brain networks," *Frontiers in Neuroscience*, vol. 4, p. 200, 2010.

[79] J. Mihm, C. H. Loch, D. Wilkinson, and B. A. Huberman, "Hierarchical structure and search in complex organizations," *Management Science*, vol. 56, no. 5, pp. 831–848, 2010.

[80] B. Mišić, R. F. Betzel, A. Nematzadeh, J. Goni, A. Griffa, P. Hagmann, A. Flammini, Y.-Y. Ahn, and O. Sporns, "Cooperative and competitive spreading dynamics on the human connectome," *Neuron*, vol. 86, no. 6, pp. 1518–1529, 2015.

[81] C. R. Myers, "Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs," *Physical Review E*, vol. 68, no. 4, p. 046 116, 2003.

[82] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[83] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[84] M. E. Newman, S. Forrest, and J. Balthrop, "Email networks and the spread of computer viruses," *Physical Review E*, vol. 66, no. 3, p. 035 101, 2002.

[85] K. Oda and H. Kitano, "A comprehensive map of the toll-like receptor signaling network," *Molecular Systems Biology*, vol. 2, no. 1, 2006.

[86] B. O. Palsson, *Systems Biology*. Cambridge university press, 2015.

[87] R. K. Pan, N. Chatterjee, and S. Sinha, "Mesoscopic organization reveals the constraints governing caenorhabditis elegans nervous system," *PloS one*, vol. 5, no. 2, e9240, 2010.

[88] D. L. Parnas, P. C. Clements, and D. M. Weiss, "The modular structure of complex systems," in *Proceedings of the 7th International Conference on Software Engineering*, IEEE Press, 1984, pp. 408–417.

[89] J. K. Pirri and M. J. Alkema, "The neuroethology of c. elegans escape," *Current opinion in neurobiology*, vol. 22, no. 2, pp. 187–193, 2012.

[90] M. Quint, H.-G. Drost, A. Gabel, K. K. Ullrich, M. Bönn, and I. Grosse, "A transcriptomic hourglass in plant embryogenesis," *Nature*, vol. 490, no. 7418, pp. 98–101, 2012.

[91] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant visual representation by single neurons in the human brain," *Nature*, vol. 435, no. 7045, pp. 1102–1107, 2005.

[92] A. Raj and Y.-h. Chen, "The wiring economy principle: Connectivity determines anatomy in the human brain," *PloS one*, vol. 6, no. 9, e14832, 2011.

[93] E. Ravasz and A.-L. Barabási, "Hierarchical organization in complex networks," *Physical Review E*, vol. 67, no. 2, p. 026 112, 2003.

[94] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[95] J. B. Reece, L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, R. B. Jackson, *et al.*, *Campbell biology*. Pearson Boston, 2014.

[96] J. Rexford and C. Dovrolis, "Future internet architecture: Clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, 2010.

[97] D. L. Riddle, T. Blumenthal, B. J. Meyer, and J. R. Priess, "Mechanosensory control of locomotion," 1997.

[98] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[99] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, "Core-periphery structure in networks," *SIAM Journal on Applied mathematics*, vol. 74, no. 1, pp. 167–190, 2014.

[100] K. M. Sabrin and C. Dovrolis, "The hourglass effect in hierarchical dependency networks," *Network Science*, vol. 5, no. 4, pp. 490–528, 2017.

[101] H. Saito, M. Toyoda, M. Kitsuregawa, and K. Aihara, "A large-scale study of link spam detection by graph algorithms," in *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, ACM, 2007, pp. 45–48.

[102] M. Sales-Pardo, R. Guimera, A. A. Moreira, and L. A. N. Amaral, "Extracting the hierarchical organization of complex systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 39, pp. 15 224–15 229, 2007.

[103] W. R. Schafer, "Mechanosensory molecules and circuits in c. elegans," *Pflügers Archiv-European Journal of Physiology*, vol. 467, no. 1, pp. 39–48, 2015.

[104] M. A. Schilling, "Toward a general modular systems theory and its application to interfirm product modularity," *Academy of Management Review*, vol. 25, no. 2, pp. 312–334, 2000.

[105] H. A. Simon, *The architecture of complexity*. Springer, 1991, pp. 457–476.

[106] P. Siyari, B. Dilkina, and C. Dovrolis, "Lexis: An optimization framework for discovering the hierarchical structure of sequential data," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1185–1194.

[107] P. Siyari, B. Dilkina, and C. Dovrolis, "Emergence and evolution of hierarchical structure in complex systems," *arXiv preprint arXiv:1805.04924*, 2018.

[108] C. Smolke, *The Metabolic Pathway Engineering Handbook: Tools and Applications*. CRC press, 2009, vol. 1.

[109] Y. Sohn, M.-K. Choi, Y.-Y. Ahn, J. Lee, and J. Jeong, "Topological cluster analysis reveals the systemic organization of the caenorhabditis elegans connectome," *PLoS computational biology*, vol. 7, no. 5, e1001139, 2011.

[110] J. Stelling, U. Sauer, Z. Szallasi, F. J. Doyle, and J. Doyle, "Robustness of cellular functions," *Cell*, vol. 118, no. 6, pp. 675–685, 2004.

[111] J. Supper, L. Spangenberg, H. Planatscher, A. Dräger, A. Schröder, and A. Zell, "Bowtiebuilder: Modeling signal transduction pathways," *BMC Systems Biology*, vol. 3, no. 1, p. 1, 2009.

[112] J. M. Swaminathan, S. F. Smith, and N. M. Sadeh, "Modeling supply chain dynamics: A multiagent approach*," *Decision Sciences*, vol. 29, no. 3, pp. 607–632, 1998.

[113] R Tanaka, M Csete, and J Doyle, "Highly optimised global organisation of metabolic networks," *IEE Proceedings-Systems Biology*, vol. 152, no. 4, pp. 179–184, 2005.

[114] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[115] E. K. Towlson, P. E. Vértes, S. E. Ahnert, W. R. Schafer, and E. T. Bullmore, "The rich club of the c. elegans neuronal connectome," *Journal of Neuroscience*, vol. 33, no. 15, pp. 6380–6387, 2013.

[116] S. Valverde and R. V. Solé, "Self-organization versus hierarchy in open-source social networks," *Physical Review E*, vol. 76, no. 4, p. 046 118, 2007.

[117] M. P. Van Den Heuvel and O. Sporns, "Rich-club organization of the human connectome," *Journal of Neuroscience*, vol. 31, no. 44, pp. 15 775–15 786, 2011.

[118] S. Varier and M. Kaiser, "Neural development features: Spatio-temporal development of the caenorhabditis elegans neuronal network," *PLoS computational biology*, vol. 7, no. 1, e1001044, 2011.

[119] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii, "Structural properties of the caenorhabditis elegans neuronal network," *PLoS computational biology*, vol. 7, no. 2, e1001066, 2011.

[120] S. Vitali, J. B. Glattfelder, and S. Battiston, "The network of global corporate control," *PloS One*, vol. 6, no. 10, e25995, 2011.

[121] G. P. Wagner, M. Pavlicev, and J. M. Cheverud, "The road to modularity," *Nature Reviews Genetics*, vol. 8, no. 12, pp. 921–931, 2007.

[122] T. Wakabayashi, I. Kitagawa, and R. Shingai, "Neurons regulating the duration of forward locomotion in caenorhabditis elegans," *Neuroscience research*, vol. 50, no. 1, pp. 103–111, 2004.

[123] D. Weinshenker, G. Garriga, and J. H. Thomas, "Genetic and pharmacological analysis of neurotransmitters controlling egg laying in c. elegans," *Journal of Neuroscience*, vol. 15, no. 10, pp. 6975–6985, 1995.

[124] K.-K. Yan, G. Fang, N. Bhardwaj, R. P. Alexander, and M. Gerstein, "Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks," *Proceedings of the National Academy of Sciences*, vol. 107, no. 20, pp. 9186–9191, 2010.

[125] H. Yu and M. Gerstein, "Genomic analysis of the hierarchical structure of regulatory networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 40, pp. 14 724–14 731, 2006.

[126] H. Yu, P. M. Kim, E. Sprecher, V. Trifonov, and M. Gerstein, "The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics," *PLoS Comput Biol*, vol. 3, no. 4, e59, 2007.

[127] J. Zhao, H. Yu, J.-H. Luo, Z.-W. Cao, and Y.-X. Li, "Hierarchical modularity of nested bow-ties in metabolic networks," *BMC Bioinformatics*, vol. 7, no. 1, p. 386, 2006.